UNIVERSIDAD MAYOR, REAL Y PONTIFICIA DE SAN FRANCISCO XAVIER DE CHUQUISACA

VICERRECTORADO

CENTRO DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

FACULTAD DE CIENCIAS Y TECNOLOGÍA

ESTRATEGIAS DE AUTOMATIZACIÓN DE PRUEBAS EN LA INTEGRACIÓN CONTINUA PARA EL DESARROLLO DE APLICACIONES WEB

TRABAJO EN OPCIÓN A DIPLOMADO EN DEVELOPMENT OPERATIONS "DEVOPS" V.1.

MIGUEL ÁNGEL LOPEZ PECHO

Sucre - Bolivia

2024

CESIÓN DE DERECHOS

Al presentar este trabajo como requisito previo para la obtención del Diploma en Development Operations "DEVOPS" V.1. de la Universidad Mayor, Real y Pontificia de San Francisco Xavier de Chuquisaca, autorizo al Centro de Estudios de Posgrado e Investigación o a la Biblioteca de la Universidad, para que se haga de este trabajo un documento disponible para su lectura según normas de la Universidad

También cedo a la Universidad Mayor, Real y Pontifica de San Francisco Xavier de Chuquisaca, los derechos de publicación de este trabajo o parte de él, manteniendo mis derechos de autor hasta un periodo de 30 meses posterior a su aprobación.

Miguel Ángel López Pecho

……..

FIRMA:

DEDICATORIA

Dedico este trabajo de grado a mis padres Manuel López Torres (†) y Cecilia Pecho Barañado Viuda de Lopez y a toda mi familia, que con mucho amor, respaldo me han ayudado con su apoyo incondicional en todo momento, por ser ellos mi principal motor y estímulo para seguir siempre mirando hacia adelante.

AGRADECIMIENTO

Agradecimiento a los docentes de la Carrera de Ingeniería de Sistemas, Facultad de tecnología de la Universidad de San Francisco Xavier de Chuquisaca, al Centro de Estudios de Posgrado e Investigación por sus enseñanzas, por el tiempo y la dedicación y por todos sus conocimientos que me ayudaron en la realización de esta monografía.

Al apoyo incondicional de mis padres y familiares por el apoyo que siempre me brindaron a lo largo de todos mis estudios.

A todos los amigos que de alguna forma han contribuido con sus ideas y apoyo al momento de realizar esta investigación.

RESUMEN

La monografía titulada "Estrategias de Automatización de Pruebas en la Integración Continua para el Desarrollo de Aplicaciones Web" aborda la importancia de la automatización de pruebas en el contexto de la Integración Continua (CI) para optimizar el desarrollo de aplicaciones web. El objetivo general es proponer un conjunto de estrategias más efectivas de automatización de pruebas en este marco.

En el desarrollo de la monografía, se aborda inicialmente el marco teórico, que incluye una introducción al desarrollo de aplicaciones web y los principios de DevOps, así como conceptos fundamentales de la Integración Continua y la automatización de pruebas. Se analizan diversas estrategias y mejores prácticas de automatización de pruebas, incluidas las pruebas unitarias, de integración y de aceptación, junto con métricas relevantes para evaluar su efectividad.

La metodología de investigación utilizada es explicativa, con un enfoque mixto que combina métodos teorico y empíricos. Se utilizan técnicas como el estudio de casos y la observación de documentos para recopilar datos y analizar las prácticas actuales de desarrollo de aplicaciones web en relación con la automatización de pruebas.

Los objetivos específicos incluyen definir las estrategias y herramientas de automatización de pruebas, valorar su eficacia y eficiencia, comparar las estrategias más efectivas y proponer recomendaciones para su implementación y gestión. Se identifican limitaciones y desafíos asociados con la implementación de estas estrategias, y se analiza su impacto en la detección temprana de errores y en la confiabilidad de las aplicaciones desarrolladas.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

1. Antecedentes y Justificación

1.1. Antecedentes

Durante las últimas décadas, el desarrollo de aplicaciones web ha experimentado una notable evolución, impulsada principalmente por avances tecnológicos y cambios en las preferencias de los usuarios. Inicialmente, las aplicaciones web se limitaban principalmente a páginas estáticas con funcionalidades básicas de navegación. Sin embargo, con el advenimiento de tecnologías como HTML, CSS y JavaScript, las aplicaciones web comenzaron a ofrecer experiencias más interactivas y dinámicas, permitiendo a los usuarios realizar tareas complejas directamente desde sus navegadores.

A medida que la demanda de aplicaciones web más sofisticadas y funcionales creció, surgieron nuevos enfoques y metodologías de desarrollo para abordar los desafíos inherentes a este proceso. Los desarrolladores comenzaron a adoptar prácticas ágiles y metodologías de desarrollo iterativas para mejorar la eficiencia y la velocidad de entrega. (Celi Parraga, Bone Andrade y Mora, 2023)

Sin embargo, a medida que las aplicaciones web se volvían más complejas y críticas para las operaciones comerciales, surgieron desafíos en términos de eficiencia, calidad y velocidad de entrega, este proceso de desarrollo enfrenta desafíos inherentes, como la falta de colaboración entre equipos de desarrollo y operaciones, la complejidad en la gestión de infraestructuras y la presión por lanzamientos más frecuentes. (Rengifo, 2023)

La metodología DevOps ha surgido como un enfoque revolucionario en el mundo del desarrollo de software, uniendo a los equipos de desarrollo y operaciones para trabajar juntos y entregar valor de manera continua. Este enfoque se ha popularizado especialmente en entornos agiles y orientados a la entrega de software, donde la eficiencia y la calidad son fundamentales (Felipe Redondo, Nuñez Cardenas, 2022).

Además, DevOps fomenta una mentalidad de colaboración y responsabilidad compartida entre los equipos de desarrollo y operaciones. Los desarrolladores y los administradores de sistemas trabajan juntos desde las primeras etapas del ciclo de vida del desarrollo de software, lo que permite identificar y solucionar problemas de manera más rápida y eficiente. Esta colaboración continua garantiza que las necesidades y preocupaciones de

ambas partes se aborden durante todo el proceso de desarrollo, mejorando así la calidad y estabilidad de la aplicación web final. (Aranguren Velasquez y Ruiz Pedraza, 2023)

Uno de los aspectos clave de la aplicación de DevOps en el desarrollo de aplicaciones web es la automatización de procesos. Esto incluye la automatización de la integración continua (CI), la entrega continua (CD) y las pruebas automatizadas, lo que permite a los equipos de desarrollo desplegar cambios en el código de manera rápida y segura (Sharma y Coyne, 2015).

El uso de herramientas y técnicas de automatización de pruebas en el contexto de la Integración Continua y la Entrega Continua es fundamental para garantizar la calidad del software en el desarrollo de aplicaciones web. La automatización de pruebas permite detectar errores de manera temprana, facilita la identificación de problemas de integración y mejora la confiabilidad del software en entornos de desarrollo ágiles y dinámicos (AppMaster, 2023).

Según Tom Hall en su artículo Prácticas recomendadas de DevOps indica:

> Una transformación a DevOps requiere una revisión de las estructuras y los procesos empresariales, pero el esfuerzo vale la pena. En nuestra encuesta sobre tendencias de DevOps de 2020, el 99 % de los encuestados declaró que DevOps tuvo un impacto positivo en su organización. Otro estudio sobre DevOps, el Informe del estado de DevOps de 2019 realizado por DORA, reveló que los profesionales de élite publican con una frecuencia 208 superior y 106 veces más rápido que los equipos de bajo rendimiento. Además, no se trata solo de la velocidad de salida al mercado: DevOps ofrece una calidad mejorada y propicia que los equipos de élite tengan una tasa de fallos de cambio siete veces inferior en comparación con los equipos de bajo rendimiento. (Hall, 2024)

En Bolivia, el concepto de DevOps está comenzando a ganar relevancia, aunque las empresas del sector público aún no han establecido un proceso de desarrollo seguro. Sin embargo, conforme se introducen términos relacionados con DevOps y seguridad en los lineamientos y estándares para el desarrollo de software de alta calidad en entidades gubernamentales, se está avanzando en este aspecto. Estos lineamientos han sido

elaborados por el grupo de trabajo de desarrollo de software del consejo para las tecnologías de información y comunicación del estado plurinacional de Bolivia. Actualmente, están en proceso de aprobación por parte de la Agencia de Gobierno Electrónico y Tecnologías de Información y Comunicación (AGETIC) y el Consejo para las Tecnologías de Información y Comunicación del Estado Plurinacional de Bolivia (CTIC-EPB). Después de su aprobación inicial, serán sometidos a una segunda evaluación por parte de las entidades públicas y, finalmente, serán publicados para que todas las entidades públicas cumplan con estos lineamientos (Pachacuti Blanco, 2020).

1.2. Justificación

El desarrollo de aplicaciones web se ha convertido en un elemento esencial para empresas y organizaciones en la actualidad, ya que les permite ofrecer servicios y productos de forma eficiente y escalable a través de Internet. Sin embargo, el proceso de desarrollo de aplicaciones web enfrenta desafíos significativos en términos de velocidad, calidad y eficiencia. La adopción de prácticas DevOps, centradas en la Integración Continua y la Entrega Continua, ofrece una solución viable para abordar estos desafíos al promover la colaboración entre equipos y la automatización de procesos en todo el ciclo de vida del desarrollo de software (Rengifo, 2023).

La implementación efectiva de estrategias de automatización de pruebas en el contexto de la CI y de la CD es fundamental para garantizar la calidad del software en el desarrollo de aplicaciones web. La automatización de pruebas permite detectar errores de manera temprana, facilita la identificación de problemas de integración y mejora la confiabilidad de del software en entornos de desarrollo agiles y dinámicos (Autentia, 2023).

Estas estrategias son fundamentales en el desarrollo de software mediante la implementación de metodologías ágiles. Algunos de los beneficios que proporciona su uso son:

- Se mantiene al día con los cambios continuos.
- Reduce posibles errores manuales.
- Proporciona un proceso repetitivo y acelera la llegada del producto al cliente, sin intervención humana.

La implementación de prácticas DevOps y la automatización de pruebas en el desarrollo de aplicaciones web pueden generar importantes beneficios económicos para las organizaciones. La reducción del tiempo de desarrollo, la detección temprana de errores y la mejora de la calidad del software pueden ayudar a reducir los costos asociados con el desarrollo de aplicaciones web, así como a aumentar la satisfacción del cliente y la competitividad en el mercado. Además, la inversión en herramientas y tecnologías de automatización de pruebas puede considerarse como una inversión a largo plazo que puede generar un retorno significativo en términos de eficiencia y calidad del desarrollo de software (Valdez Valda, 2022).

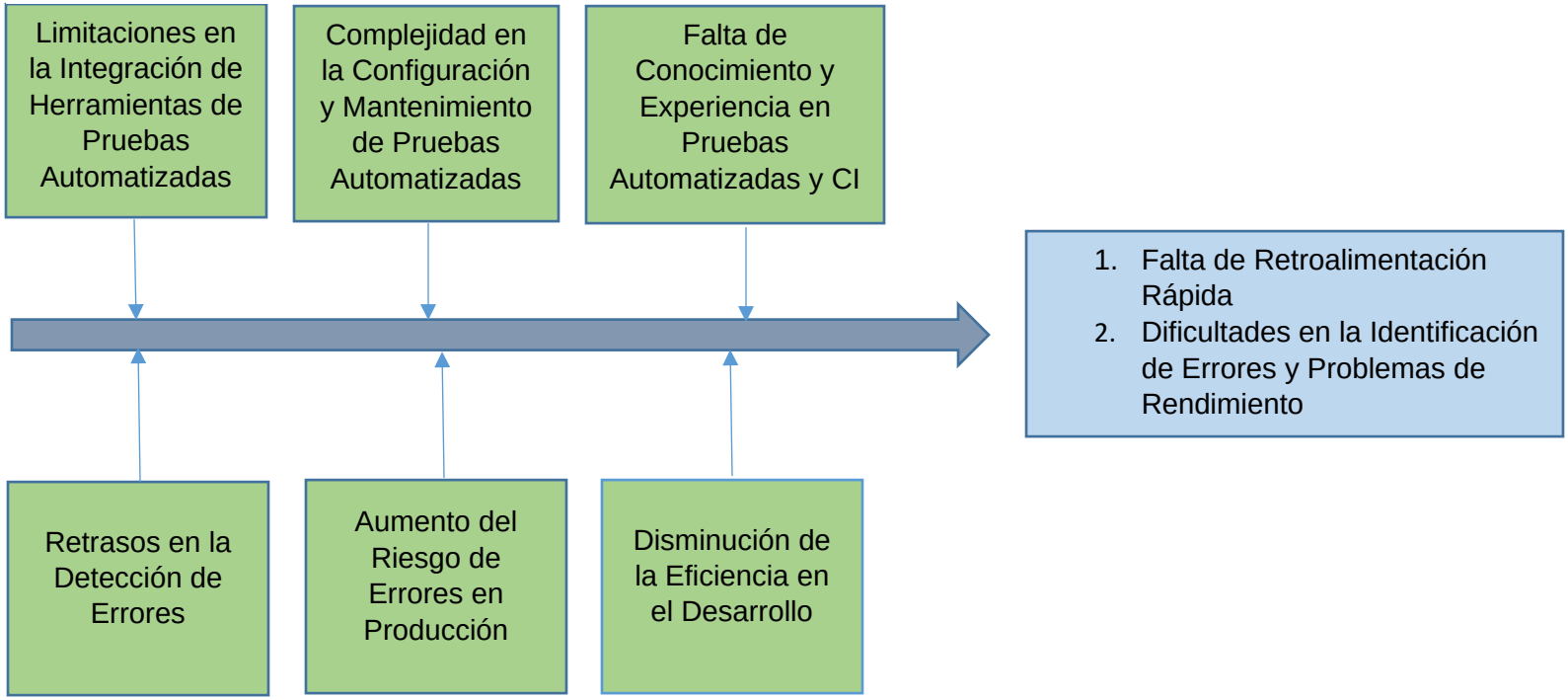
La implementación efectiva de DevOps puede conducir a una reducción de costos operativos y una mayor rentabilidad. Según Puppet's State of DevOps Report (2019), las organizaciones que adoptan DevOps experimentan una reducción del 22% en el tiempo dedicado a trabajo no planificado y una disminución del 50% en el tiempo de recuperación ante incidentes. Estos ahorros en tiempo y recursos pueden traducirse directamente en una reducción de costos operativos. (PUPPET, CIRCLECI, SPLUNK, 2019)

2. Situación Problémica

2.1. Diagrama Causa - Efecto

Figura 1

Diagrama Causa - Efecto



Nota: Elaboración Propia

2.2. Identificación del problema

La implementación de prácticas DevOps en el desarrollo de aplicaciones web también contribuye significativamente al control de calidad. Al automatizar las pruebas y la implementación, los equipos pueden identificar y abordar errores y problemas en etapas iniciales del ciclo de desarrollo. Esto disminuye el riesgo de errores en producción y asegura que las actualizaciones de software mantengan altos estándares de calidad.

Entre Las principales causas que podemos identificar tenemos:

- Limitaciones en la Integración de Herramientas de Pruebas Automatizadas y CI, la falta de integración entre herramientas de pruebas automatizadas y sistemas de CI dificulta la ejecución automática de pruebas durante el proceso de construcción y despliegue, lo que resulta en una menor eficiencia y confiabilidad en la detección de errores.
- Complejidad en la Configuración y Mantenimiento de Pruebas Automatizadas, la configuración y mantenimiento de suites de pruebas automatizadas puede ser compleja y requiere un esfuerzo significativo, lo que dificulta su incorporación efectiva en los flujos de trabajo de CI y aumenta el riesgo de errores y fallos.
- Falta de Conocimiento y Experiencia en Pruebas Automatizadas y CI, la falta de capacitación y experiencia en el uso de herramientas de pruebas automatizadas y CI entre los equipos de desarrollo puede dificultar la implementación efectiva de estrategias de automatización de pruebas en el proceso de desarrollo de aplicaciones web.

Lo que conlleva a los siguientes efectos:

- Retrasos en la Detección de Errores, la falta de integración adecuada entre herramientas de pruebas automatizadas y CI puede ocasionar retrasos en la detección de errores, lo que impacta en la velocidad de entrega y la calidad del software desarrollado.
- Aumento del Riesgo de Errores en Producción, la falta de pruebas automatizadas integradas en los flujos de trabajo de CI aumenta el riesgo de errores y fallos en producción, lo que puede resultar en costos adicionales y una menor satisfacción del cliente.

- Disminución de la Eficiencia en el Desarrollo, la complejidad en la configuración y mantenimiento de pruebas automatizadas puede disminuir la eficiencia en el proceso de desarrollo, aumentando los costos y retrasando la entrega de software.

Esto nos ocasiona problemas como:

- Falta de Retroalimentación Rápida, la falta de integración entre pruebas automatizadas y CI dificulta la obtención de retroalimentación rápida sobre la calidad del software desarrollado, lo que dificulta la toma de decisiones informadas durante el proceso de desarrollo.
- Dificultades en la Identificación de Errores y Problemas de Rendimiento, la falta de pruebas automatizadas integradas en los flujos de trabajo de CI puede dificultar la identificación temprana de errores y problemas de rendimiento, lo que aumenta el riesgo de fallos en producción y una menor satisfacción del usuario.

3. Formulación del problema de investigación

¿Cuáles son las estrategias más efectivas de automatización de pruebas en la Integración Continua para mejorar la calidad y eficiencia en el desarrollo de aplicaciones web?

4. Objetivo General

Proponer un conjunto de estrategias efectivas de automatización de pruebas en el marco de la Integración Continua para optimizar el desarrollo de aplicaciones web.

5. Objetivos Específicos

- Investigar las diferentes estrategias de automatización de pruebas utilizadas en la Integración Continua para el desarrollo de aplicaciones web.
- Describir las principales herramientas utilizadas en la automatización de pruebas para la Integración Continua en el desarrollo de aplicaciones web.
- Valorar la eficacia y eficiencia de cada estrategia de automatización de pruebas en términos de detección de errores, cobertura de pruebas y tiempo de ejecución recurriendo a documentación oficial.

6. Diseño Metodológico

6.1. Tipo de Investigación

La metodología utilizada es la de Investigación Explicativa, ya que se entra en la etapa de análisis y explicación de los fenómenos observados, buscando comprender las relaciones causales entre las variables involucradas en nuestro estudio. Es decir, se investiga cómo la implementación de estrategias de automatización de pruebas en la Integración Continua (CI) afecta la eficiencia y la calidad del desarrollo de aplicaciones web. Se emplea un enfoque de investigación mixta que combina enfoques cualitativos y cuantitativos.

Para la parte cualitativa, se emplean técnicas como informes en profundidad o análisis de contenido para explorar las percepciones, opiniones y experiencias de los participantes en relación con la automatización de pruebas en la Integración Continua.

Para la parte cuantitativa, se recopilan datos numéricos sobre el rendimiento de las pruebas automatizadas, la cobertura de código, el tiempo de ejecución, entre otros, para cuantificar el impacto de las estrategias de automatización en el proceso de desarrollo de aplicaciones web.

6.1.1. Alcance de la investigación

El alcance de la monografía sobre "Estrategias de Automatización de Pruebas en la Integración Continua para el Desarrollo de Aplicaciones Web" abarca una investigación de buenas prácticas utilizadas en la automatización de pruebas en entornos de desarrollo web. La monografía se centra en proponer un conjunto de estrategias que pueden ser utilizadas para integrar de manera efectiva en los procesos de Integración Continua, con el objetivo de mejorar la calidad, eficiencia y velocidad de entrega del software.

Es importante destacar que la investigación se centra en la conceptualización y aplicación de estas estrategias en el contexto de la Integración Continua, evitando detalles técnicos especializados y la implementación detallada de soluciones. Con este alcance, se busca ofrecer una visión integral y práctica de cómo la automatización de pruebas puede mejorar significativamente los procesos de desarrollo de aplicaciones web en entornos ágiles y orientados a DevOps.

6.2. Métodos de Investigación

6.2.1. Métodos Teóricos

6.2.1.1. Método sistémico

Este método implica la descripción del marco teórico, ya que describimos la arquitectura y los componentes de una aplicación web desde el enfoque de desarrollo con DevOps. Además, utilizamos el método sistémico para analizar cómo los diferentes componentes del proceso de desarrollo de software (como las herramientas de automatización de pruebas, los sistemas de integración continua, los equipos de desarrollo, etc.) interactúan entre sí y cómo estas interacciones afectan el rendimiento general del sistema.

6.2.1.2. Método histórico lógico

Este método se utiliza para estudiar eventos pasados y su evolución a lo largo del tiempo. Se empleó el método histórico en la sección de antecedentes, donde investigamos cómo ha evolucionado la automatización de pruebas en el contexto de la Integración Continua para el desarrollo de aplicaciones web a lo largo de los años. Esto nos permitió comprender mejor el contexto histórico lógico y las tendencias que han dado forma al tema de investigación.

6.2.1.3. Método de Análisis y síntesis

Durante la elaboración de la monografía, se recurrió ampliamente al uso de este método, ya que resulta sumamente útil para la búsqueda y el manejo de la información, incluyendo tareas como clasificación, desglose y división de contenidos. Se aplicó en varias etapas del proceso, incluyendo la delimitación del tema, la redacción de la situación problemática, la formulación del problema, la redacción de los objetivos y la elaboración del marco teórico.

6.2.1.4. Método de Análisis Documental

Este método implica la recopilación y análisis de documentos relevantes, como libros, artículos, informes técnicos, normativas, entre otros. Se utilizara el análisis documental en varias secciones de la monografía, incluyendo la revisión de la literatura, donde se identificara y revisara investigaciones previas y recursos relacionados con la automatización de pruebas en la Integración Continua. También se podrá emplear este método para recopilar y analizar documentación técnica sobre herramientas, metodologías y prácticas relacionadas con el tema de investigación.

6.2.2. Métodos Empíricos

6.2.2.1. Método Observación

Al analizar documentos implícitamente está el método de Observación. El método de observación de documentos implica el análisis detallado de documentos relevantes relacionados con la integración continua y la automatización de pruebas en el desarrollo de aplicaciones web.

6.3. Técnicas

6.3.1. Observación

La observación de documentos es una técnica de investigación cualitativa que implica el análisis y la interpretación de documentos escritos, electrónicos o multimedia relevantes para el tema de estudio.

6.3.2. Análisis documental

Se recopilaron y analizaron documentos técnicos, informes de proyectos, registros de desarrollo de software y otros materiales relevantes para identificar tendencias, patrones y prácticas comunes relacionadas con la automatización de pruebas en la Integración Continua. El análisis de documentos ayudó a complementar y respaldar argumentos con evidencia concreta.

6.4. Procedimientos e instrumentos de investigación

6.4.1. Reportes o Informes (Encuetas)

Los reportes o informes son herramientas fundamentales para evaluar el progreso, identificar áreas de mejora y tomar decisiones informadas dentro de cualquier proyecto o empresa. En nuestro caso específico, nos referimos a los informes anuales generados por la encuesta DORA (DevOps Research and Assessment), una fuente invaluable de datos y análisis en el ámbito de la ingeniería de software y las prácticas DevOps.

La encuesta DORA es reconocida internacionalmente por su enfoque riguroso en la evaluación de las prácticas de desarrollo de software, las cuales abarcan desde la implementación de cambios hasta la frecuencia y la estabilidad de las entregas. Estos informes proporcionan una visión detallada del estado del arte en términos de cultura, procesos y herramientas dentro de las organizaciones que adoptan metodologías DevOps.

6.4.2. Análisis de documentos

Utilizamos técnicas de búsqueda bibliográfica para identificar y recopilar investigaciones previas, libros, artículos y otros recursos relevantes sobre la automatización de pruebas en la Integración Continua. La revisión de la literatura ayudó a contextualizar el estudio y a comprender el estado actual del conocimiento en el campo.

6.4.3. Lista de Cotejo

La lista de cotejo es una herramienta que te permite evaluar y registrar la presencia o ausencia de ciertos criterios o características en documentos, observaciones o eventos. En nuestro caso tendría el objetivo de evaluar la calidad y relevancia de los documentos observados en relación con la automatización de pruebas y la Integración Continua.

CAPITULO I

MARCO TEÓRICO Y CONTEXTUAL

1.1. Marco Conceptual

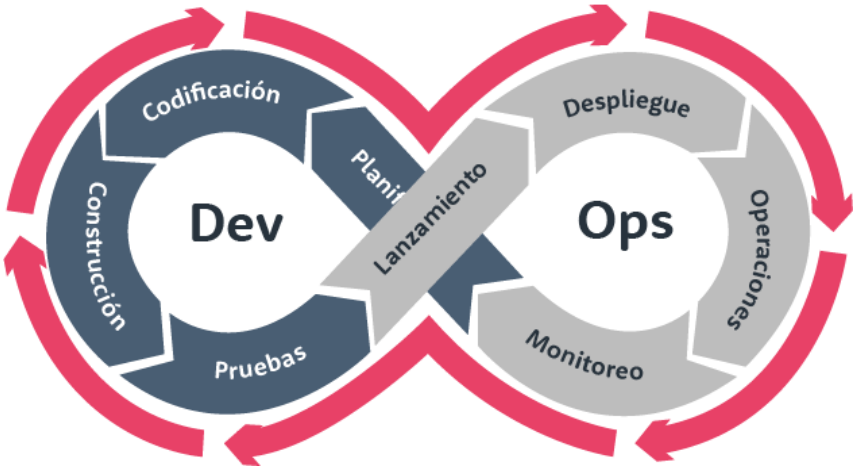
1.1.1. Development Operations DEVOPS

DevOps es una combinación de las abreviaturas "Development" (desarrollo) y " Operations" (operaciones). Podríamos definirlo como un marco de trabajo en el que tanto el departamento de desarrollo como el de operaciones colaboran aportando ideas, pruebas prácticas y el uso de nuevas tecnologías para conseguir mejorar los procesos de desarrollo del software. En ningún caso, se trata de aumentar la responsabilidad de los desarrolladores, haciéndoles responsables de funciones del departamento de operaciones. (Autentia, 2023).

La metodología DevOps representa un método para el desarrollo de software que fusiona las áreas de desarrollo y operaciones con el objetivo de establecer un proceso de entrega de software eficaz, fiable y cooperativo. Se fundamenta en la premisa de que los equipos de desarrollo y operaciones deben cooperar de manera unificada y colaborativa a lo largo de todo el ciclo de vida del software.

Figura 2

Ciclo de Vida de una Aplicación



Nota: Adaptado de *Representación Gráfica de DEVOPS*, por SENTRIO, 2021, SENTRIO (https://sentrio.io/blog/que-es-devsecops-vs-devops/)

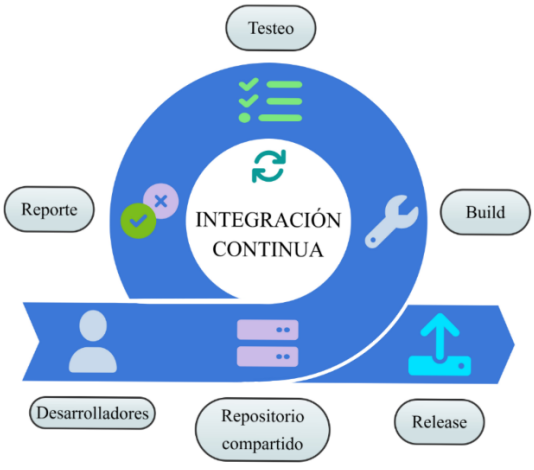
1.1.2. Integración Continua CI

La Integración Continua (CI) es un proceso clave en el desarrollo de software, donde los desarrolladores utilizan herramientas de control de versiones para almacenar y controlar las distintas versiones de un proyecto. Esto facilita el acceso a la historia y avances del proyecto para todos los involucrados, permitiendo también la medición del progreso del equipo a través de sprints, ciclos de trabajo definidos en metodologías ágiles. Los sprints, con una duración generalmente no superior a dos semanas, permiten realizar cambios significativos sin abrumar al equipo, facilitando la identificación y corrección temprana de errores. La Integración Continua busca minimizar errores sin sacrificar la calidad ni aumentar el tiempo de desarrollo con correcciones (Cruz Gonzales y Franco Calderon, 2023).

La Integración Continua es una práctica de desarrollo de software en la que los desarrolladores combinan su trabajo en un repositorio compartido de forma regular, generalmente varias veces al día. Cada combinación se verifica automáticamente mediante la compilación automatizada (incluidas pruebas) para detectar errores de integración lo más rápido posible. Esta práctica permite a los equipos detectar y solucionar problemas de forma más rápida y eficiente, lo que conduce a un proceso de desarrollo más fluido y confiable.

Figura 3

Integración Continua



Nota: Adaptado de *Integraciones continuas ¿Qué son y para que sirven?*, por Marianna Rolfo, 2022, codigoencasa.com (https://codigoencasa.com/integraciones-continuas-que-son-y-para-que-sirven/)

1.1.3. Entrega Continua CD

La Entrega Continua (continuous delivery) está relacionada con la integración continua y consiste en la automatización del proceso de entrega del software, permitiendo que pueda ser implementado en producción de forma confiable y sencilla. De forma práctica se puede entender la CD como la entrega de actualizaciones de software a los usuarios o clientes de forma sólida y continua. (Autentia, 2023).

La Entrega Continua es una extensión de la integración continua en la que el software se puede poner en producción de manera segura y confiable en cualquier momento. La idea es que el código esté siempre en un estado desplegable y listo para ser implementado en producción. Esto implica la automatización de las pruebas, la construcción y el despliegue del software, así como la implementación de un conjunto robusto de prácticas y herramientas para garantizar la calidad y la confiabilidad del proceso de entrega.

1.1.4. Despliegue Continuo CD

El Despliegue Continúo (continuous deployment) es la capacidad de llevar cambios de todo tipo (incluyendo nuevas funcionalidades, cambios de configuración, corrección de errores...) al entorno de producción o a manos de los usuarios finales de forma segura, lo más automática posible, rápida y continua. El objetivo es hacer despliegues predecibles, automáticos y rutinarios en cualquier momento, ya sea de un sistema distribuido a gran escala, un entorno de producción complejo, un sistema integrado o una aplicación. (Autentia, 2023).

El Despliegue Continuo lleva la idea de la entrega continua un paso más allá al automatizar completamente el proceso de implementación del software en producción. Con el despliegue continuo, cualquier cambio que pase las pruebas y la validación automáticamente se implementa en producción sin intervención humana. Esto permite a las organizaciones lanzar cambios de software de manera rápida y segura, lo que resulta en una mayor agilidad y capacidad de respuesta a las necesidades del mercado.

1.1.5. Diferencia entre entrega Continúa y Despliegue continuo

La diferencia principal entre la Entrega Continua y el Despliegue Continuo radica en el último paso del proceso. Mientras que la Entrega Continua garantiza que el software esté siempre listo para ser entregado en producción, el Despliegue Continuo va un paso más

allá al automatizar completamente la implementación del software en producción, eliminando la necesidad de intervención humana en el proceso de despliegue. (openinnova, 2022)

Figura 4

Entrega Continua vs Despliegue Continuo

Nota: Adaptado de *Entrega Continua vs Despliegue Continuo*, por Qualizens Trainin Provider, 2021, Qualizens (http://training.qualizens.com/dou)

1.1.6. Automatización de Pruebas

La práctica de automatización de pruebas consiste en revisar y validar automáticamente un producto de software (por ejemplo, una aplicación web) para asegurarse de que cumple con los estándares de calidad predefinidos para el estilo de código, la funcionalidad (lógica empresarial) y la experiencia del usuario. (Hristov, 2024)

La automatización de pruebas es una estrategia fundamental en el desarrollo de software, que implica el uso de herramientas y scripts para ejecutar pruebas de forma automatizada en lugar de realizarlas manualmente. Esta práctica permite una verificación rápida y precisa del comportamiento del software, ayudando a identificar errores y garantizar la calidad del producto final. Al eliminar la necesidad de realizar pruebas repetitivas de forma manual, la automatización de pruebas ahorra tiempo y recursos, y facilita la detección temprana de problemas, lo que resulta en un proceso de desarrollo más eficiente y un software más fiable.

1.1.7. Aplicaciones Web

Una aplicación web (web-based application) es un tipo especial de aplicación cliente/servidor, donde tanto el cliente (el navegador, explorador o visualizador1) como el servidor (el servidor web) y el protocolo mediante el que se comunican (HTTP) están estandarizados y no han de ser creados por el programador de aplicaciones. (Lujan Mora, 2017)

Las aplicaciones web son programas informáticos diseñados para ser utilizados a través de un navegador web, sin necesidad de instalación en el dispositivo del usuario. Estas aplicaciones se ejecutan en un servidor remoto y se accede a ellas mediante una conexión a Internet. Permiten a los usuarios interactuar con servicios, acceder a información, realizar transacciones y realizar diversas tareas a través de una interfaz web. Las aplicaciones web pueden variar en complejidad y funcionalidad, desde simples páginas estáticas hasta aplicaciones dinámicas y sofisticadas que ofrecen experiencias interactivas en tiempo real. Ejemplos comunes de aplicaciones web incluyen servicios de correo electrónico, redes sociales, tiendas en línea y herramientas de productividad en la nube.

1.2. Marco Teórico

1.2.1. Introducción al Desarrollo de Aplicaciones Web y DevOps

1.2.1.1. Breve descripción del desarrollo de aplicaciones web y los principios de DevOps.

En la actualidad, el desarrollo de aplicaciones web se ha convertido en un componente fundamental de la estrategia digital de muchas organizaciones. Este enfoque de desarrollo se centra en la creación de aplicaciones que funcionan a través de navegadores web, permitiendo a los usuarios acceder y utilizar servicios y funciones desde cualquier dispositivo con conexión a Internet. Las aplicaciones web abarcan una amplia gama de funcionalidades, desde simples sitios web estáticos hasta complejas plataformas de comercio electrónico y aplicaciones de software-as-a-service (SaaS).

Por otro lado, DevOps, es una metodología que busca integrar los equipos de desarrollo de software y operaciones de TI para mejorar la colaboración, la eficiencia y la calidad del software. Los principios fundamentales de DevOps incluyen la automatización de procesos, la entrega continua, la monitorización y la retroalimentación constante. Esta filosofía promueve la idea de un ciclo de desarrollo de software más rápido y eficiente, donde los equipos trabajan de manera conjunta para ofrecer valor al cliente de manera rápida y sostenible (AppMaster, 2023).

1.2.2. Fundamentos de la Integración Continua

1.2.2.1. Principios básicos

La Integración Continua es una práctica de desarrollo de software que se centra en la automatización y la entrega frecuente de cambios en el código. Se basa en la idea de integrar regularmente los cambios realizados por los desarrolladores en un repositorio compartido, lo que permite una detección temprana de errores y una rápida retroalimentación sobre la calidad del código. Este enfoque se apoya en principios como la automatización de pruebas, la integración frecuente, el despliegue continuo y el control de versiones, que buscan mantener el código siempre en un estado desplegable y garantizar un flujo de trabajo eficiente y colaborativo.

Al automatizar procesos como la construcción del código y la ejecución de pruebas, la CI permite a los equipos desarrollar software de manera más ágil y confiable. Proporciona una retroalimentación rápida sobre la calidad del código y la integridad del sistema, lo que facilita

la detección temprana de problemas y su corrección oportuna. Además, al utilizar un entorno de pruebas similar al de producción y practicar la entrega continua, la CI contribuye a la reducción de riesgos y errores en el proceso de desarrollo, al tiempo que acelera la entrega de nuevas funcionalidades y mejoras a los usuarios finales (Autentia, 2023).

1.2.2.2. Beneficios y objetivos

La adopción de la Integración Continua conlleva una serie de beneficios, entre los que se incluyen (Autentia, 2023):

- Detección temprana de errores: Al integrar los cambios de manera frecuente y automatizada, los errores se detectan rápidamente, lo que facilita su resolución y evita que se propaguen en el código base.
- Mayor calidad del software: La ejecución de pruebas automatizadas en cada integración garantiza que el software se mantenga funcional y libre de errores en todo momento.
- Aceleración del ciclo de desarrollo: La automatización de procesos como la compilación, las pruebas y el despliegue permite acelerar el ciclo de desarrollo y entrega de software.
- Mejora de la colaboración: La CI fomenta una cultura de colaboración entre los miembros del equipo, ya que todos trabajan sobre la misma base de código y se aseguran de que los cambios se integren de manera fluida.

1.2.2.3. Componentes principales de la Integración Continua

Los componentes principales de la Integración Continua incluyen (Autentia, 2023):

- Repositorio de código: Un sistema de control de versiones donde se almacena el código fuente del proyecto y se realizan los cambios.
- Servidor de CI: Una plataforma que automatiza el proceso de integración y ejecución de pruebas en respuesta a cada cambio en el repositorio de código.
- Herramientas de construcción y pruebas: Utilidades que se encargan de compilar el código, ejecutar pruebas automatizadas y generar informes de resultados.
- Notificaciones y alertas: Mecanismos que informan al equipo sobre el estado de las integraciones y los resultados de las pruebas, facilitando la detección y resolución de problemas.

1.2.3. Automatización de pruebas en la Integración Continua

La automatización de pruebas en la Integración Continua es un componente fundamental para garantizar la calidad del software en el desarrollo de aplicaciones web. Esta práctica consiste en la creación y ejecución automatizada de pruebas durante el proceso de Integración Continua, donde los cambios de código se integran frecuente y automáticamente en un entorno compartido. La finalidad principal de la automatización de pruebas en este contexto es detectar rápidamente errores o regresiones en el software, lo que ayuda a mantener la estabilidad y funcionalidad del sistema en todo momento. (Jet Brains, 2000-2024)

1.2.3.1. Tipos de pruebas automatizadas

La automatización de pruebas en la Integración Continua se apoya en herramientas y frameworks especializados que permiten escribir, ejecutar y analizar pruebas de manera automatizada. Estas pruebas pueden abarcar diferentes niveles, como pruebas unitarias, pruebas de integración, pruebas de aceptación y pruebas de rendimiento entre otras, y se ejecutan de forma continua cada vez que se realiza una integración de código. Esto proporciona retroalimentación inmediata sobre la calidad del software, permitiendo a los equipos identificar y solucionar problemas de manera ágil. (PARASOFT, 2024)

1.2.4. Mejores prácticas de automatización de pruebas

Las pruebas de integración continua son un aspecto crítico del proceso de desarrollo de software que ayuda a garantizar la calidad y confiabilidad de su software. Al seguir estas prácticas se puede crear un canal de entrega de software sólido y confiable que le permita entregar software de alta calidad de manera rápida y eficiente (FasterCapital, 2024). Podemos Indicar las siguientes:

1.2.4.1. Selección de pruebas a automatizar

La selección de pruebas a automatizar es un paso crucial en el proceso de automatización de pruebas. No todas las pruebas son adecuadas para la automatización, por lo que es importante identificar aquellas que proporcionarán el mayor valor y retorno de inversión. En general, las pruebas que son repetitivas, propensas a errores humanos y críticas para la estabilidad y funcionalidad del sistema son buenas candidatas para la automatización. Estas suelen incluir pruebas de regresión, pruebas de integración y pruebas de aceptación.

1.2.4.2. Creación de conjuntos de pruebas robustos

La creación de conjuntos de pruebas robustos es esencial para garantizar la eficacia de la automatización de pruebas. Esto implica diseñar casos de prueba que sean claros, concisos, completos y mantenibles. Es importante cubrir diferentes escenarios de uso y casos límite para garantizar una cobertura adecuada. Además, se deben evitar dependencias entre casos de prueba para que puedan ejecutarse de forma independiente. Utilizar técnicas como la parametrización y la reutilización de casos de prueba puede ayudar a mantener conjuntos de pruebas más robustos y flexibles.

1.2.4.3. Gestión de datos de prueba y generación de informes

La gestión de datos de prueba y la generación de informes son aspectos importantes de la automatización de pruebas. Los datos de prueba deben ser consistentes, relevantes y fácilmente accesibles para garantizar la reproducibilidad de las pruebas. Esto puede implicar el uso de herramientas de gestión de datos de prueba o la implementación de bases de datos de prueba dedicadas. Además, es crucial generar informes detallados sobre los resultados de las pruebas, incluyendo información sobre los casos de prueba ejecutados, los errores encontrados y la cobertura de pruebas alcanzada. Estos informes son fundamentales para evaluar la calidad del software y tomar decisiones informadas sobre su entrega.

1.2.5. Estrategias de automatización de pruebas

Si bien existen diferentes tipos de pruebas nos enfocamos en dos estrategias de Pruebas. Las pruebas unitarias y pruebas de integración dentro del contexto de la integración continua ya que ofrecen una combinación poderosa que garantiza una cobertura exhaustiva, eficiencia en la detección de errores y cumplimiento de requisitos, todo ello con una reducción significativa de riesgos.

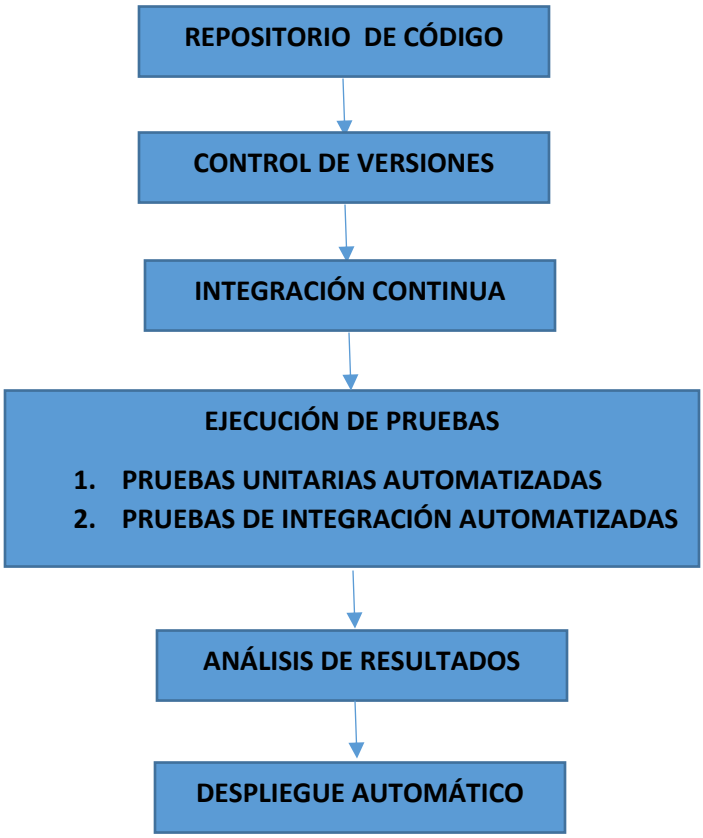
- Las pruebas unitarias automatizadas se enfocan en verificar el correcto funcionamiento de unidades individuales de código, como métodos o funciones, de manera aislada. Estas pruebas se ejecutan de forma automatizada cada vez que se realiza una modificación en el código, lo que permite detectar rápidamente posibles errores y garantizar que las nuevas funcionalidades no introduzcan regresiones en el sistema. (Giraldo Colorado y Giraldo Plaza, Julio-Diciembre de 2013)

- Por otro lado, las pruebas de integración automatizadas se centran en verificar la interacción entre diferentes componentes o módulos del sistema, asegurando que funcionen correctamente en conjunto. Estas pruebas son fundamentales en entornos de integración continua, ya que garantizan la estabilidad y la cohesión del sistema a medida que se realizan cambios en el código base. (QAlified, 2023)

1.2.5.1. Flujo de trabajo de Integración Continua con Pruebas Automatizadas: Garantizando Calidad y Eficiencia en el Desarrollo de Software

Figura 5

Flujo de trabajo de Integración Continua con Pruebas Automatizadas



Nota: Elaboración Propia

Este esquema ilustra cómo las pruebas automatizadas son parte integral del proceso de Integración Continua, ayudando a garantizar la calidad del software en cada etapa del desarrollo. En este esquema:

- El código fuente se almacena en un repositorio de control de versiones, como por ejemplo Jenkins, Travis CI, CircleCI o GitLab CI.
- A través del control de versiones, se envía el código al sistema de Integración Continua (CI).
- En el CI, se ejecutan automáticamente las pruebas unitarias, de integración y de aceptación.
- Después de ejecutar las pruebas, se analizan los resultados para determinar si los cambios cumplen con los criterios de calidad establecidos.
- Si las pruebas pasan con éxito, el código puede ser desplegado automáticamente en el entorno de producción.

En este esquema, el proceso de Integración Continua (CI) se muestra como un flujo de trabajo automatizado que facilita la entrega continua de software de alta calidad. Comienza con el almacenamiento del código fuente en un repositorio de control de versiones, desde donde se envía al sistema de CI para su integración. Una vez allí, el código se somete a una serie de pruebas automatizadas, incluyendo pruebas unitarias y de integración. Estas pruebas son esenciales para identificar y corregir posibles errores en el código de manera temprana, antes de su despliegue en un entorno de producción.

El análisis de los resultados de las pruebas es una etapa crítica en este proceso, ya que proporciona retroalimentación inmediata sobre la calidad del software. Si todas las pruebas pasan con éxito, el código puede avanzar automáticamente al siguiente paso, que es el despliegue automatizado en el entorno de producción. Este enfoque garantiza una entrega rápida y confiable de nuevas características y correcciones de errores, al tiempo que reduce el riesgo de introducir fallos en el sistema en producción. En resumen, el esquema de Integración Continua con pruebas automatizadas es fundamental para mantener un flujo de trabajo eficiente y una alta calidad del software en todo momento.

1.2.5.2. Pruebas Unitarias Automatizadas

1.2.5.2.1. Definición y objetivo de las pruebas unitarias.

Las pruebas unitarias son un tipo de prueba automatizada que se centra en verificar el funcionamiento individual de unidades específicas de código, como funciones, métodos o clases, de forma aislada. El objetivo principal de las pruebas unitarias es garantizar que cada unidad de código funcione correctamente según lo esperado y produzca los resultados deseados. Al probar unidades de código de forma aislada, las pruebas unitarias permiten detectar y corregir errores en una etapa temprana del ciclo de desarrollo, lo que contribuye a mejorar la calidad del software y a reducir el tiempo y los costos asociados con la corrección de errores más adelante en el proceso. (Rehkopf, 2024)

1.2.5.2.2. Herramientas y tecnologías para la automatización de pruebas unitarias.

Existen numerosas herramientas y tecnologías disponibles para la automatización de pruebas unitarias en el desarrollo de aplicaciones web. Algunas de las herramientas más populares incluyen JUnit para Java, NUnit para .NET, PHPUnit para PHP y Jasmine para JavaScript. Estas herramientas proporcionan marcos de prueba y bibliotecas que permiten a los desarrolladores escribir, ejecutar y analizar pruebas unitarias de manera eficiente. Además, las herramientas de construcción y administración de proyectos, como Apache Maven y Gradle, pueden integrarse con estas herramientas de prueba para automatizar completamente el proceso de ejecución de pruebas unitarias como parte de un flujo de trabajo de Integración Continua. (Hiberus, 2023)

- JUnit para Java: JUnit es un framework de pruebas unitarias para el lenguaje de programación Java. Permite escribir y ejecutar pruebas de manera automatizada para validar el comportamiento de clases y métodos en un proyecto Java. JUnit proporciona anotaciones y aserciones que facilitan la creación de pruebas claras y concisas. (Jakubiak, 2022).
- NUnit para .NET: NUnit es un framework de pruebas unitarias para aplicaciones desarrolladas en el ecosistema .NET, incluyendo C#, Visual Basic .NET, y F#. Al igual que JUnit, NUnit permite escribir y ejecutar pruebas de manera automatizada, proporcionando una sintaxis similar que hace que sea fácil para los desarrolladores migrar entre ambos entornos. (Gonzalez Espinoza y Bello Lara, 2016).

- PHPUnit para PHP: PHPUnit es una herramienta para realizar pruebas unitarias en aplicaciones PHP. Es ampliamente utilizado en el desarrollo de software PHP, permitiendo a los desarrolladores escribir pruebas para clases y métodos de forma eficiente. PHPUnit ofrece un conjunto completo de funcionalidades para escribir pruebas robustas y realizar pruebas de integración y de extremo a extremo. (Geek, 2023).
- Jasmine para JavaScript: Jasmine es un framework de pruebas para JavaScript, especialmente diseñado para realizar pruebas en el lado del cliente y en el lado del servidor utilizando Node.js. Jasmine utiliza una sintaxis basada en funciones de JavaScript para definir y ejecutar pruebas de forma intuitiva. Es adecuado para realizar pruebas de comportamiento (BDD) y pruebas unitarias en aplicaciones web y móviles desarrolladas con JavaScript. (Tejada Yau, 2024)

1.2.5.3. Pruebas de Integración Automatizadas

1.2.5.3.1. Concepto y propósito de las pruebas de integración.

Las pruebas de integración son un tipo de prueba que se centra en verificar la interacción y el funcionamiento conjunto de diferentes componentes o módulos de un sistema. El propósito principal de estas pruebas es asegurar que los diversos componentes de una aplicación funcionen correctamente juntos cuando se integran. Esto implica probar la comunicación entre los componentes, la transferencia adecuada de datos y la interoperabilidad entre las diferentes partes del sistema. Las pruebas de integración son cruciales para identificar errores de integración y problemas de comunicación que pueden surgir cuando se combinan varios módulos en una aplicación más grande. (Rehkopf, 2024)

1.2.5.3.2. Herramientas y tecnologías para la automatización pruebas de integración.

Las herramientas y tecnologías para la automatización de pruebas de integración son fundamentales para garantizar que los diferentes componentes de un sistema funcionen correctamente juntos. Estas herramientas facilitan la ejecución de pruebas que verifican la interacción y la interoperabilidad entre los diversos módulos o servicios de una aplicación. Algunas de las herramientas más utilizadas para la automatización de pruebas de integración incluyen (Hiberus, 2023):

- Selenium WebDriver: Es una herramienta de automatización de pruebas diseñada específicamente para pruebas de interfaz de usuario web. Permite a los desarrolladores escribir scripts de prueba en varios lenguajes de programación, como Java, Python, C#, Ruby, etc. Selenium WebDriver interactúa con el navegador web de la misma manera que lo haría un usuario real, lo que permite realizar acciones como hacer clic en botones, completar formularios, navegar por páginas y verificar elementos de la interfaz de usuario. Es altamente flexible y se puede integrar fácilmente con diferentes herramientas de desarrollo y marcos de trabajo de prueba. (QAlified, 2022).
- Docker: Docker es una plataforma de contenedores que permite empaquetar, distribuir y ejecutar aplicaciones en entornos aislados llamados contenedores. En el contexto de la automatización de pruebas dentro de la integración continua, Docker se utiliza para crear entornos de pruebas consistentes y reproducibles. Los contenedores Docker encapsulan todas las dependencias y configuraciones necesarias para ejecutar las pruebas, lo que garantiza que los resultados de las pruebas sean coherentes independientemente del entorno de ejecución. Esto facilita la configuración y el despliegue de pruebas automatizadas en diferentes entornos, lo que mejora la eficiencia del proceso de desarrollo de software. (Fernandez, 2023)
- SoapUI: SoapUI es una herramienta especializada en pruebas de servicios web basados en el protocolo SOAP. Permite crear, ejecutar y administrar pruebas de integración para servicios web SOAP y RESTful. SoapUI ofrece capacidades de automatización y generación de informes detallados sobre el rendimiento y la interoperabilidad de los servicios. (Garcia Puebla, 2009)
- JUnit: Aunque JUnit es ampliamente conocido por sus capacidades de pruebas unitarias en Java, también se puede utilizar para pruebas de integración. JUnit proporciona un marco de pruebas robusto y flexible que permite a los desarrolladores escribir y ejecutar pruebas que verifican la integración entre diferentes componentes de un sistema. (Jakubiak, 2022).

Las pruebas unitarias y de integración tienen propósitos distintos y se llevan a cabo en varias fases del ciclo de vida de desarrollo. La siguiente tabla compara las diferencias entre las pruebas unitarias y las pruebas de integración:

Tabla 1

Diferencias entre Pruebas Unitarias y Pruebas de Integración

Aspecto	Pruebas Unitarias	Pruebas de integración
Alcance	Se concentran en unidades o módulos particulares.	Evaluan la interacción de módulos integrados.
Objetivo	Comprobar que cada elemento funciona de forma independiente.	Comprobar que todas las piezas conectadas entre sí funcionan correctamente.
Interoperabilidad	Las dependencias externas se mantienen separadas de las pruebas.	Se requiere la integración con módulos del mundo real.
Velocidad	Puesto que se concentra en unidades pequeñas, la ejecución es más rápida.	El tiempo de ejecución es un poco mayor debido a las pruebas de los diversos módulos.
Cobertura	Ofrece cobertura extensiva de código al inspeccionar módulos de forma exhaustiva.	Garantiza que los módulos funcionen correctamente como elemento del sistema genera.
Flujo de Trabajo de las Pruebas	Normalmente, los desarrolladores realizan pruebas unitarias antes que las pruebas de integración.	Estas suelen llevarse a cabo durante la fase de integración del software.

Nota: Adaptado de *Pruebas Unitarias vs. Pruebas de Integración*, por QAlified, 2023, QAlified Building Quality (https://qalified.com/es/blog/pruebas-de-integracion-que-son/)

1.2.6. Implementación Práctica de Automatización de Pruebas en la Integración Continua

La implementación práctica de la automatización de pruebas en la Integración Continua para el desarrollo de aplicaciones web implica configurar un entorno adecuado, desarrollar scripts de pruebas automatizadas y ejecutar estas pruebas de manera automatizada como parte del flujo de trabajo de CI/CD (Humble y Farley, Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation, 2010).

1.2.6.1. Configuración del Entorno de Integración Continua

Para comenzar, es necesario configurar un entorno de Integración Continua adecuado. Esto generalmente implica seleccionar una plataforma de CI/CD, como Jenkins, Travis CI,

CircleCI o GitLab CI, y configurarla según las necesidades del proyecto. Se deben definir los pasos de construcción, prueba y despliegue dentro del archivo de configuración de CI/CD, asegurando que las pruebas automatizadas se ejecuten como parte de este proceso.

Además, es crucial establecer un entorno de prueba consistente que refleje el entorno de producción tanto como sea posible. Esto puede implicar el uso de contenedores Docker para crear entornos reproducibles y aislados para ejecutar pruebas automatizadas.

1.2.6.2. Desarrollo de Scripts de Pruebas Automatizadas

El siguiente paso es desarrollar scripts de pruebas automatizadas que cubran los casos de prueba relevantes para la aplicación web. Para las pruebas de unidad, se pueden utilizar frameworks de pruebas como JUnit, NUnit, Jasmine o PHPUnit, según la tecnología utilizada en el proyecto. Estos scripts deben ser capaces de ejecutarse de manera independiente y proporcionar resultados claros sobre el estado de cada prueba.

Para las pruebas de integración, los scripts deben simular la interacción entre los diferentes componentes de la aplicación. Esto puede incluir el uso de herramientas como Selenium WebDriver, SoapUI, Doker.

Es fundamental que los scripts de pruebas automatizadas sean mantenibles, modulares y escalables. Esto facilita la incorporación de nuevas pruebas a medida que la aplicación evoluciona y permite la rápida detección y corrección de errores.

1.2.6.3. Ejecución de Pruebas Automatizadas en un Entorno de CI/CD

Una vez que se han desarrollado los scripts de pruebas automatizadas, se deben integrar en el flujo de trabajo de CI/CD para que se ejecuten automáticamente en cada confirmación de código. Esto implica configurar los pasos de prueba dentro del proceso de construcción en la plataforma de CI/CD seleccionada.

Las pruebas automatizadas deben ejecutarse en un entorno controlado y aislado para garantizar la consistencia de los resultados. Los resultados de las pruebas, incluidos los casos de éxito y las fallas, deben registrarse y notificarse automáticamente para que los desarrolladores puedan tomar medidas correctivas rápidamente.

Además, es importante establecer criterios claros para la aceptación de cambios en el repositorio de código. Esto puede incluir la integración de pruebas de código, cobertura de código y resultados de pruebas automatizadas como parte de los requisitos para fusionar cambios en la rama principal del repositorio.

1.2.7. Métricas relevantes para evaluar la efectividad de las estrategias de automatización de pruebas en la Integración Continua

Las métricas desempeñan un papel fundamental en la evaluación de la efectividad de las estrategias de automatización de pruebas en la Integración Continua. Estas métricas proporcionan información valiosa sobre el rendimiento y la calidad del proceso de desarrollo de software, lo que permite a los equipos identificar áreas de mejora y tomar decisiones informadas sobre cómo optimizar sus prácticas de prueba. (Estayno, Dapozo, Cuenca Pletch, y Greiner, 2009)

1.2.7.1. Detección de errores

Medir la eficacia en la detección de errores es esencial para evaluar la robustez de las estrategias de automatización de pruebas. Contar con métricas como el número de errores detectados por caso de prueba y el tiempo medio para resolver un error permite entender cómo cada estrategia contribuye a la identificación temprana y la rápida resolución de problemas. (QAwerk, 2023)

- Número de errores detectados por caso de prueba: Esta métrica te permite conocer la efectividad de cada estrategia en la detección de errores durante la ejecución de los casos de prueba.
- Tiempo medio para resolver un error: Indica la rapidez con la que se resuelven los errores detectados durante las pruebas, lo cual puede ser crucial para mantener un desarrollo ágil y eficiente.

1.2.7.2. Cobertura de pruebas

Evaluar la cobertura de pruebas proporciona una visión integral de qué tan exhaustivas son las estrategias implementadas. Con métricas como la cobertura de código y la cobertura de funcionalidades, es posible determinar el alcance y la efectividad de las pruebas automatizadas en la validación de todas las partes críticas del sistema. (QAwerk, 2023)

- Cobertura de código: Esta métrica evalúa el porcentaje de código que ha sido ejecutado durante las pruebas automatizadas, lo que proporciona una medida de la exhaustividad de las pruebas.
- Cobertura de funcionalidades: Evalúa qué porcentaje de las funcionalidades del sistema ha sido probado, lo que garantiza que todas las características importantes estén siendo validadas.

1.2.7.3. Tiempo de ejecución:

El tiempo de ejecución de las pruebas es un factor crucial en la eficiencia del proceso de desarrollo. Al analizar métricas como el tiempo total de ejecución de pruebas y el tiempo medio por caso de prueba, se puede identificar el impacto de cada estrategia en la duración de las pruebas automatizadas y tomar medidas para optimizar su rendimiento. (QAwerk, 2023).

- Tiempo total de ejecución de pruebas: Indica cuánto tiempo tarda en ejecutarse el conjunto completo de pruebas automatizadas, lo que puede afectar la eficiencia del proceso de desarrollo.
- Tiempo medio por caso de prueba: Mide la duración promedio de la ejecución de cada caso de prueba, lo que puede ayudar a identificar pruebas que son particularmente lentas y que pueden requerir optimización.

1.3. Marco contextual

1.3.1. Contextualización del Desarrollo de Aplicaciones Web

1.3.1.1. Descripción del panorama actual del desarrollo de aplicaciones web

Para comprender el panorama actual del desarrollo de aplicaciones web, es importante considerar varios aspectos relevantes. En la actualidad, el desarrollo de aplicaciones web es un campo en constante evolución, impulsado por avances tecnológicos, cambios en las demandas de los usuarios y nuevas tendencias en el mercado.

El desarrollo de aplicaciones web en el panorama actual se caracteriza por la diversidad de tecnologías y herramientas, el enfoque en la experiencia del usuario, la agilidad en el proceso de desarrollo, el énfasis en la seguridad y privacidad, y la adopción de estrategias avanzadas de despliegue y mantenimiento. Estos aspectos deben ser considerados al

diseñar y desarrollar aplicaciones web exitosas en el contexto actual. A continuación, se describen algunos aspectos clave (Lerna-Blasco, Murcia Andres, y Mifsud Talon, 2013):

- Tecnologías y Herramientas Utilizadas, el desarrollo de aplicaciones web involucra una amplia variedad de tecnologías y herramientas, que van desde lenguajes de programación como HTML, CSS y JavaScript, hasta frameworks y plataformas de desarrollo como React, Angular, Vue.js, Django, Ruby on Rails, entre otros. Estas tecnologías y herramientas están en constante evolución, con nuevas versiones y actualizaciones que buscan mejorar la eficiencia y la experiencia del usuario.
- Tendencias en Diseño y Experiencia del Usuario, en el contexto actual, el diseño y la experiencia del usuario son aspectos fundamentales del desarrollo de aplicaciones web. Las tendencias en diseño web incluyen enfoques minimalistas, interfaces intuitivas, diseño responsivo para adaptarse a diferentes dispositivos y técnicas de micro interacciones para mejorar la interacción del usuario. Además, la accesibilidad web y la inclusión de tecnologías como Progressive Web Apps (PWAs) están ganando relevancia para ofrecer una experiencia de usuario óptima.
- Metodologías de Desarrollo y Prácticas Ágiles, el desarrollo de aplicaciones web se beneficia cada vez más de metodologías ágiles como Scrum, Kanban y Lean, que permiten una entrega rápida, iterativa y flexible de software. Estas metodologías fomentan la colaboración entre equipos multidisciplinarios, la retroalimentación continua y la adaptación a los cambios en los requisitos del proyecto.
- Seguridad y Privacidad, con el aumento de las amenazas cibernéticas, la seguridad y la privacidad son aspectos críticos en el desarrollo de aplicaciones web. Las prácticas de seguridad incluyen el uso de HTTPS, la autenticación de usuarios, la protección contra ataques de inyección de código y la gestión adecuada de datos sensibles para garantizar la confidencialidad e integridad de la información.
- Estrategias de Despliegue y Mantenimiento, la implementación y el mantenimiento de aplicaciones web requieren estrategias eficientes de despliegue y gestión de infraestructura. La adopción de enfoques como la integración continua (CI), la entrega continua (CD) y la automatización de tareas de despliegue facilita la gestión del ciclo de vida del software y garantiza la disponibilidad y estabilidad de las aplicaciones.

1.3.1.2. Evolución histórica del desarrollo de aplicaciones web y su impacto en la necesidad de estrategias de automatización de pruebas

El impacto de esta evolución en la necesidad de estrategias de automatización de pruebas es evidente. Con aplicaciones web cada vez más complejas y dinámicas, el proceso de prueba manual se vuelve insostenible en términos de tiempo y recursos. La automatización de pruebas se ha vuelto esencial para garantizar la calidad del software en un entorno de desarrollo ágil y DevOps, permitiendo pruebas rápidas y repetibles que se integran perfectamente en los flujos de trabajo de desarrollo y despliegue. Se destacan algunos de los puntos clave en esta evolución (Lujan Mora, 2017):

- Nacimiento de la Web Estática, en los primeros días de la web, las páginas web eran estáticas y se componían principalmente de texto e imágenes simples. El desarrollo web estaba centrado en la creación de contenido estático y la presentación básica de información en línea.
- Introducción de Tecnologías Dinámicas, con el avance de las tecnologías de servidor y cliente, surgieron herramientas y lenguajes como PHP, ASP, y JavaScript, lo que permitió la creación de sitios web dinámicos y interactivos. Esto abrió la puerta a aplicaciones web más complejas y funcionales.
- Auge de las Aplicaciones Web Interactivas, en la década de 2000, con la popularización de AJAX (Asynchronous JavaScript and XML), las aplicaciones web comenzaron a volverse más interactivas y responsivas. Surgieron aplicaciones como Gmail y Google Maps, que ofrecían experiencias de usuario similares a las aplicaciones de escritorio.
- Adopción de Frameworks y Arquitecturas Modernas, en los últimos años, hemos visto la adopción generalizada de frameworks y arquitecturas modernas como React, Angular, Vue.js, y arquitecturas de microservicios. Estas tecnologías permiten el desarrollo de aplicaciones web más escalables, modulares y mantenibles (Lerna-Blasco, Murcia Andres, y Mifsud Talon, 2013).
- Crecimiento del Desarrollo Ágil y DevOps, la adopción de metodologías ágiles y prácticas de DevOps ha transformado la forma en que se desarrollan y despliegan las aplicaciones web. La entrega continua (CD) y la integración continua (CI) se han vuelto estándares en la industria, lo que ha acelerado el ritmo de desarrollo y aumentado la presión para mantener altos niveles de calidad (Lerna-Blasco, Murcia Andres, y Mifsud Talon, 2013).

1.3.1.3. Tendencias Actuales en el desarrollo de aplicaciones web, como la adopción de prácticas DevOps

Las tendencias actuales en el desarrollo de aplicaciones web están marcadas por la adopción generalizada de prácticas DevOps, que han transformado la forma en que se planifica, desarrolla y despliega el software en la industria. Aquí hay una descripción más detallada de estas tendencias (Lerna-Blasco, Murcia Andres, y Mifsud Talon, 2013):

- Prácticas DevOps busca integrar el desarrollo y las operaciones en un ciclo de vida de desarrollo de software continúo, ha ganado una amplia aceptación en la industria. Las prácticas DevOps promueven la colaboración estrecha entre los equipos de desarrollo y operaciones, la automatización de procesos, la entrega continua y la monitorización continua del rendimiento del software en producción.
- La adopción de la integración continua (CI) y la entrega continua (CD) se ha convertido en un componente central de las prácticas DevOps. La CI implica la automatización de la compilación y prueba de código cada vez que se realiza una actualización en el repositorio de código, mientras que la CD se refiere a la automatización del proceso de entrega de software al entorno de producción de manera rápida y confiable.
- Microservicios y Arquitecturas Basadas en Contenedores, la tendencia hacia arquitecturas basadas en microservicios y contenedores (como Docker y Kubernetes) ha permitido a las organizaciones descomponer sus aplicaciones en componentes más pequeños y manejables. Esto facilita el desarrollo, despliegue y escalado independiente de diferentes partes de la aplicación, lo que resulta en una mayor agilidad y flexibilidad.

1.3.2. Desarrollo de Aplicaciones Web: Impacto Social, Económico y Cultural

1.3.2.1. Impacto Social

- Las aplicaciones web han transformado la forma en que las personas interactúan y se comunican en la sociedad moderna.
- Han facilitado el acceso a la información, la educación, los servicios de salud, el entretenimiento y las redes sociales, contribuyendo a la democratización del conocimiento y la conectividad.
- Las aplicaciones web también han impulsado el surgimiento de nuevas formas de colaboración, trabajo remoto y participación ciudadana en línea. (AppMaster, 2023)

1.3.2.2. Impacto Económico

- El desarrollo de aplicaciones web ha generado una industria próspera y dinámica, creando empleo y oportunidades económicas en todo el mundo.
- Ha facilitado la creación y la expansión de empresas digitales, startups y emprendimientos innovadores, impulsando el crecimiento económico y la competitividad.
- Las aplicaciones web también han transformado los modelos de negocio tradicionales, permitiendo la venta en línea, la publicidad digital, los servicios de suscripción y otros modelos de monetización. (AppMaster, 2023)

1.3.2.3. Impacto Cultural

- Las aplicaciones web han influido en la forma en que las personas consumen medios, acceden al entretenimiento y participan en la cultura digital.
- Han fomentado la diversidad cultural y el intercambio global de ideas, arte y expresiones culturales a través de plataformas en línea.
- Sin embargo, también han planteado desafíos en términos de privacidad, seguridad y manipulación de la información, lo que ha llevado a debates sobre la ética y la responsabilidad en el desarrollo y uso de aplicaciones web. (AppMaster, 2023)

CAPITULO II

2. Diagnostico

2.1. Introducción

En el capítulo del diagnóstico de nuestra monografía, nos adentramos en la aplicación práctica de las estrategias, herramientas y técnicas descritas en los apartados anteriores. Este capítulo representa el punto culminante de nuestra investigación, donde pondremos a prueba los objetivos y los planteamientos teóricos mediante la aplicación de métodos empíricos y análisis detallados. A través de esta fase, buscamos evaluar la efectividad de las estrategias de automatización de pruebas en la Integración Continua y sugerir recomendaciones prácticas para optimizar el proceso de desarrollo de aplicaciones web.

2.1.1. Procesamiento y Análisis de Datos

2.1.1.1. Diseño de Instrumentos (Análisis Documental)

Tabla 2

Matriz de Categorización de Estrategias de Automatización de Pruebas en la Integración Continua

Categoría / Variable	Sub categorías / Dimensiones	Indicadores
Estrategias de automatización de pruebas en la Integración Continua	Tipos de estrategias	Estrategias de pruebas unitarias
		Estrategias de pruebas de integración
Herramientas utilizadas	Principales herramientas	Herramientas de automatización de pruebas Unitarias (por ejemplo: Junit (Java), Nunit (.NET), PHPUnit (PHP), Jasmine (Javascript))
		Herramientas de automatización de pruebas de Integración (por ejemplo: Selenium Webdriver, Docker, SoupUI, JUnit)
Eficiencia y eficacia	Detección de errores	Número de errores detectados durante el proceso de desarrollo
	Cobertura de pruebas	Porcentaje de código cubierto por pruebas automatizadas
	Tiempo de ejecución	Tiempo promedio de ejecución de pruebas

Nota: Elaboración Propia

La matriz presenta una organización estructurada de las categorías, subcategorías y los indicadores relevantes relacionados con las estrategias de automatización de pruebas en el contexto de la Integración Continua. En la primera columna, se identifican las categorías principales, que incluyen las estrategias de automatización de pruebas, las herramientas utilizadas y la eficiencia y eficacia de estas estrategias. Cada una de estas categorías se desglosa en subcategorías o dimensiones específicas, como los tipos de estrategias de pruebas, las principales herramientas utilizadas y los indicadores clave de desempeño.

En la segunda columna, se detallan las subcategorías o dimensiones relacionadas con cada categoría principal. Por ejemplo, bajo las estrategias de automatización de pruebas, se incluyen subcategorías como estrategias de pruebas unitarias y de integración. En la tercera columna, se enumeran los indicadores específicos que se utilizan para evaluar el desempeño de cada subcategoría. Estos indicadores abarcan aspectos como la detección de errores, la cobertura de pruebas y el tiempo de ejecución de las pruebas.

En resumen, esta matriz proporciona una estructura clara y organizada para analizar y comprender las diferentes dimensiones relacionadas con la automatización de pruebas en la Integración Continua, lo que facilita la evaluación y comparación de estrategias y herramientas en el desarrollo de aplicaciones web.

2.1.1.2. Tabla comparativa de Herramientas

La tabla comparativa de herramientas utilizadas en las pruebas automatizadas de Integración y pruebas unitarias proporciona una visión estructurada y detallada de las opciones disponibles para nuestro equipo. En ella, se analizan criterios clave como la facilidad de uso, la compatibilidad con los lenguajes de programación utilizados en nuestro proyecto, la robustez de las funcionalidades de pruebas, la escalabilidad y la integración con nuestros sistemas de integración continua. Esta comparativa nos permite tomar decisiones informadas al seleccionar las herramientas más adecuadas para nuestras necesidades específicas, garantizando así una estrategia de pruebas automatizadas eficiente y efectiva que contribuya a la calidad y estabilidad de nuestro software.

Describiremos las diferentes características de las herramientas que hemos investigado para cada estrategia:

Tabla 3

Tabla comparativa de herramientas de automatización de Pruebas Unitarias

Características	JUnit	NUnit	PHPUnit	Jasmine
Lenguaje	Java	.NET (C#)	PHP	JavaScript
Tipo de Prueba	Pruebas Unitarias	Pruebas Unitarias	Pruebas Unitarias	Pruebas Unitarias
Integración CI/CD	Compatible	Compatible	Compatible	Compatible
Framework	Si	Si	Si	Si
Facilidad de Uso	Alta	Alta	Alta	Alta
Comunidad	Amplia	Amplia	Amplia	Amplia
Documentación	Abundante	Abundante	Abundante	Abundante
Soporte	Activo	Activo	Activo	Activo
Flexibilidad	Media	Media	Alta	Alta
Popularidad	Muy popular	Popular	Popular	Popular

Nota: Elaboración Propia (Especificaciones de las características de páginas oficiales de JUnit, NUnit, PHPUnit, Jasmine)

Tabla 4

Tabla comparativa de herramientas de automatización de Pruebas de Integración

Características	Selenium Webdriver	Docker	SoupUI	JUnit
Tipo de Herramienta	Pruebas de Interfaz Web	Plataforma de Contenedores	Pruebas de Servicio Web	Framework de pruebas unitarias
Interfaz Grafica	No	No	Si	No
Soporte Multiplataforma	Si	Si	Si	Si
Soporte MultiNavegador	Si	No	No	No
Automatización	Si	Si	Si	Si
Flexibilidad	Alta	Alta	Alta	Media
Documentación	Buena	Buena	Buena	Buena

Nota: Elaboración Propia (Especificaciones de las características de las paginas oficiales de Selenium Webdriver, Docker, SoupUI, JUnit)

También podemos mencionar otra tabla comparativa vista desde otro enfoque:

Tabla 5

Tabla comparativa de herramientas de automatización de Pruebas Unitarias

Características	JUnit	NUnit	PHPUnit	Jasmine
Facilidad de Configuración	Alta	Moderada	Moderada	Alta
Compatibilidad con Frameworks	Si	Si	Si	Si
Rapidez de Ejecución	Alta	Alta	Moderada	Alta
Amplia Gama de Asertos	Si	Si	Si	Si
Integración con Herramientas de CI/CD	Jenkins	Jenkins	Jenkins	Jenkins
	TravisCI	DevOps Azure	TravisCI	CircleCI
Soporte para Mocking	Si	Si	Si	Si
Compatibilidad con Principales Sistemas Operativos	Windows	Windows	Windows	Windows
	macOS	-	macOS	macOS
	Linux	Linux	Linux	Linux
Curva de Aprendizaje	Moderada	Moderada	Moderada	Moderada

Nota: Elaboración Propia (Especificaciones de las características de páginas oficiales de JUnit, NUnit, PHPUnit, Jasmine)

Tabla 6

Tabla comparativa de herramientas de automatización de Pruebas de Integración

Características	Selenium Webdriver	Docker	SoupUI	JUnit
Facilidad de Configuración	Moderada	Alta	Moderada	Moderada
Compatibilidad con Frameworks	Si	Si	No	Si
Rapidez de Ejecución	Moderada	Alta	Moderada	Alta
Amplia Gama de Asertos	Si	No	Si	Si
Integración con Herramientas de CI/CD	Jenkins	Jenkins	Jenkins	Jenkins
	TravisCI CircleCI	GitLab	TeamCity	TravisCI
Soporte para Mocking	No	No	No	Si
Compatibilidad con Principales Sistemas Operativos	Windows	Windows	Windows	Windows
	macOS	macOS	macOS	macOS
	Linux	Linux	Linux	Linux
Curva de Aprendizaje	Moderada	Moderada	Moderada	Moderada

Nota: Elaboración Propia (Especificaciones de las características de las paginas oficiales de Selenium Webdriver, Docker, SoupUI, JUnit)

2.1.1.3. Tablas comparativas de las herramientas de las pruebas automatizadas con respecto a las métricas

Tabla 7

Herramientas de Pruebas Unitarias con respecto a métricas

Métricas	JUnit	NUnit	PHPUnit	Jasmine
Detección de Errores	Alta	Alta	Alta	Media
Cobertura de Pruebas	Alta	Alta	Alta	Media
Tiempo de ejecución	Rápido	Rápido	Rápido	Rápido

Nota: Elaboración Propia (Especificaciones de las características de páginas oficiales de JUnit, NUnit, PHPUnit, Jasmine)

Tabla 8

Herramientas de Pruebas de Integración con respecto a métricas

Métricas	Selenium Webdriver	Docker	SoupUI	JUnit
Detección de Errores	Media	Baja	Alta	Alta
Cobertura de Pruebas	Media	Baja	Alta	Alta
Tiempo de ejecución	Moderado	Rápido	Moderado	Rápido

Nota: Elaboración Propia (Especificaciones de las características de las paginas oficiales de Selenium Webdriver, Docker, SoupUI, JUnit)

2.1.1.4. Lista de Cotejo de las herramientas

Esta lista nos ayudaría a garantizar que estamos evaluando adecuadamente cada herramienta en función de los criterios predefinidos, tales como funcionalidades clave, requisitos técnicos, compatibilidad con nuestra infraestructura existente y costos asociados. Al marcar cada criterio conforme se cumpla, podemos visualizar de manera clara y objetiva qué herramienta se ajusta mejor a nuestras necesidades y restricciones, facilitando así la toma de decisiones informadas y minimizando el riesgo de pasar por alto aspectos importantes durante el proceso de selección.

Tabla 9

Lista de Cotejo Herramientas de Pruebas Unitarias

Criterios	JUnit	NUnit	PHPUnit	Jasmine
Facilidad de uso	✓	✓	✓	✓
Compatibilidad con los lenguajes de programación	Java	:NET (C#, VB.NET)	PHP	JavaScript
Robustez de las funcionalidades de pruebas	✓	✓	✓	✓
Escabilidad	✓	✓	✓	✓
Integración con sistemas de integración continua	✓	✓	✓	✓
Costo	Gratis	Gratis	Gratis	Gratis
Soporte de la comunidad	✓	✓	✓	✓
Documentación disponible	✓	✓	✓	✓
Capacidad de ejecución de pruebas en paralelo	✓	✓	✓	✓
Extensibilidad y capacidad de personalización	✓	✓	✓	✗
Herramientas de desarrollo y depuración integradas	✓	✓	✓	✗

Nota: Elaboración Propia (Especificaciones de las características de páginas oficiales de JUnit, NUnit, PHPUnit, Jasmine)

Tabla 10

Lista de Cotejo Herramientas de Pruebas de Integración

Criterios	Selenium Webdriver	Docker	SoupUI	JUnit
Facilidad de uso	✓	✓	✓	✓
Compatibilidad con los lenguajes de programación	✗	✗	✗	Java
Robustez de las funcionalidades de pruebas	✓	✓	✓	✓
Escabilidad	✓	✓	✓	✓
Integración con sistemas de integración continua	✓	✓	✓	✓
Costo	Gratis	Gratis	Gratis	Gratis
Soporte de la comunidad	✓	✓	✓	✓
Documentación disponible	✓	✓	✓	✓
Capacidad de ejecución de pruebas en paralelo	✓	✓	✓	✓
Extensibilidad y capacidad de personalización	✓	✓	✓	✗
Herramientas de desarrollo y depuración integradas	✗	✗	✗	✓

Nota: Elaboración Propia (Especificaciones de las características de las paginas oficiales de Selenium Webdriver, Docker, SoupUI, JUnit)

2.1.1.5. Reportes o informes

Ante la limitación de encontrar más información sobre la automatización de pruebas en la etapa de la integración Continua en diferentes plataformas o la realización de encuetas por falta de tiempo. Nos inclinaremos a dar los beneficios dela entrega continua en el desarrollo de Software. Para ellos utilizaremos los reportes anuales de la encuesta DORA (DevOps Research and Assessment) 2023, ya que serían los más actuales ya abarcan temas de los años anteriores.

Esto nos permitirá identificar tendencias, áreas de mejora y mejores prácticas que puedan ser adaptadas para optimizar nuestros procesos y resultados. Además, estos informes nos ayudarán a establecer metas realistas y medibles, alineadas con los estándares y las mejores prácticas. Al basar nuestras decisiones en datos sólidos provenientes de fuentes confiables como la encuesta DORA, podremos avalar de manera continua y sostenible nuestra investigación de manera eficiente y confiable.

La investigación de este año 2023 exploro áreas claves para el desarrollo de Software mediante DevOps, basados en áreas como:

- Desempeño de la organización
- El rendimiento del equipo
- Bienestar de los empleados
- Rendimiento de entrega de software
- Desempeño operacional

Nos enfocaremos en el área de Rendimiento de entrega de Software. Para ello nos indican que debemos tomar en cuenta la siguiente resultado clave:

Desbloqueo de la entrega de software con revisiones de código más rápidas. Lo cual implica Optimizar el proceso de revisión de código emerge como una estrategia crucial para potenciar el desempeño del software. Equipos que agilizan estas revisiones experimentan un aumento del 50% en la eficiencia de sus entregas de software.

En cuanto a los conceptos y medidas que DORA utiliza podemos indicar que tras evaluar sus métodos de medición se escalan las puntuaciones de 0 a 10, donde 0 representa la ausencia total de un concepto y 10 que representa la máxima presencia de un concepto.

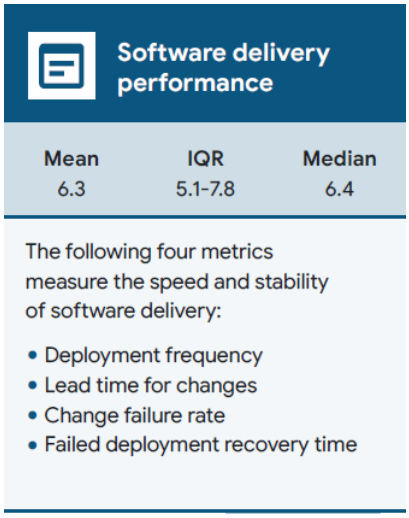
Cada concepto que se trata va acompañado con la siguiente información:

- Un icono para ayudar a transmitir el significado y facilitar su localización al utilizar este capítulo como referencia.
- La puntuación media de este concepto en la muestra (la media).
- Los límites del rango intercuartílico (IQR). Al darle los dos números (25 % y 75%) en los que se sitúa el 50% la media de los datos. Estos límites deberían ayudar a la dispersión de las respuestas.
- El valor medio de un conjunto de datos (la mediana). Si es muy diferente de la media, puede indicar que los datos son asimétricos y sesgados.
- Una descripción del concepto y de cómo medirlo.

Los resultados clave son los objetivos que creemos que las personas, los equipos o las organizaciones se esfuerzan por alcanzar (rendimiento organizativo, por ejemplo) o evitar (agotamiento, por ejemplo). En consecuencia, creemos que las medidas son formas importantes de que las personas se evalúen a sí mismas, a sus equipos y a sus organizaciones. En nuestro caso nos enfocaremos en:

Figura 6

Software delivery performance



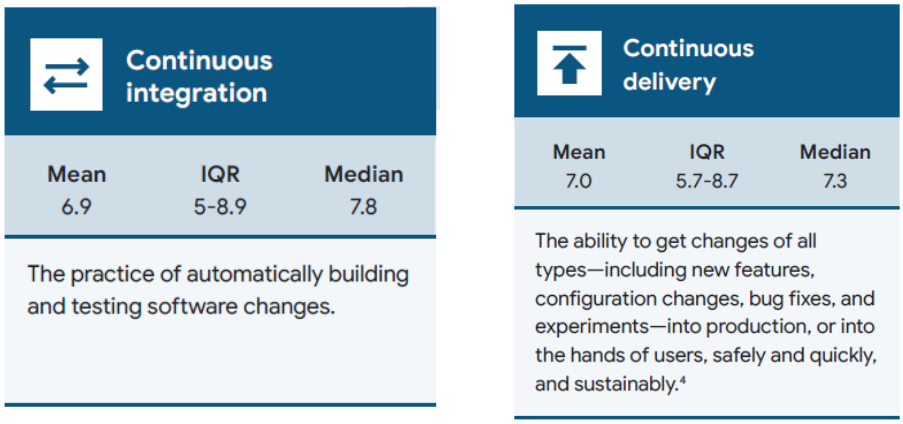
Nota: Adaptado de *Key outcomes - Software delivery performance*, por (DORA (DevOps Research and Assessment), 2023)

Donde El rendimiento de Entrega del software está basada en cuatro métricas siguientes que miden la velocidad y estabilidad de la entrega de software:

- Frecuencia de despliegue
- Tiempo de espera para los cambios
- Tasa de fallos en los cambios Operativa
- Tiempo de recuperación de una implantación fallida

Para poder lograrlo nos basaremos en procesos y sus capacidades técnicas, la cual se basa en actividades, prácticas o estados que pueden surgir en un equipo u organización. Dicho de otro modo, son cosas que hacen los equipos o formas de ser de los equipos. En la área de CI/CD

Figura 7
Continuous Integration - Continuous delivery



Nota: Adaptado de *Processes & technical capabilities – Continuous Integration – Continuous delivery*, por (DORA (DevOps Research and Assessment), 2023)

Donde en la Integración Continua nos habla acerca de la práctica de crear y probar cambios de software automáticamente. Y en la entrega contigua sobre La capacidad de poner cambios de todo tipo (incluidas nuevas funciones, cambios de configuración, correcciones de errores y experimentos) en producción o en manos de los usuarios, de forma segura, rápida y sostenible.

La siguiente figura muestra cómo diferentes capacidades y procesos técnicos pueden afectar el rendimiento del equipo, el rendimiento organizacional, el rendimiento en la entrega de software y el rendimiento operativo. Cada celda indica el nivel de impacto esperado en cada área de rendimiento para cada capacidad o proceso:

Figura 8

Results - View into how this year's survey respondents are doing with software delivery performance

Technical capabilities and processes	Effect on team performance	Effect on organizational performance	Effect on software delivery performance	Effect on operational performance
AI	No effect demonstrated	Minor increase	Minor decrease	Substantial decrease
Continuous integration	Minor increase	Minor increase	Minor increase	No effect demonstrated
Code review speed	Minor increase	No effect demonstrated	Substantial increase	Substantial increase
Loosely coupled architecture	Substantial increase	Substantial increase	Minor increase	Substantial increase
Trunk-based development	Minor increase	Minor increase	Minor increase	Minor decrease

Nota: Adaptado de *Results - View into how this year's survey respondents are doing with software delivery performance,* por (DORA (DevOps Research and Assessment), 2023)

Esta tabla proporciona una evaluación de diversas capacidades técnicas y procesos en tres áreas clave de rendimiento: el equipo, la organización y la entrega de software, así como su impacto en el rendimiento operativo. Aquí está una explicación detallada de cada parte:

- Technical capabilities and processes (Capacidades técnicas y procesos): Esta columna enumera las diferentes prácticas o capacidades técnicas evaluadas, como la inteligencia artificial (AI), la integración continua, la velocidad de revisión de código, la arquitectura desacoplada y el desarrollo basado en tronco (trunk-based development).
- Effect on team performance (Impacto en el rendimiento del equipo): Esta columna indica cómo cada capacidad o proceso afecta el rendimiento del equipo. Por ejemplo, algunas prácticas pueden resultar en un aumento menor o sustancial en la eficiencia del equipo, mientras que otras pueden no mostrar ningún efecto demostrado.
- Effect on organizational performance (Impacto en el rendimiento organizacional): Aquí se evalúa cómo cada capacidad o proceso influye en el rendimiento general de la organización. Algunas prácticas pueden tener un impacto menor o sustancial

en el rendimiento organizacional, mientras que otras pueden incluso provocar una disminución sustancial.

- Effect on software delivery performance (Impacto en el rendimiento de entrega de software): Esta columna indica cómo cada capacidad o proceso afecta la eficiencia y la efectividad de la entrega de software. Por ejemplo, algunas prácticas pueden resultar en un aumento sustancial en la velocidad y calidad de las entregas, mientras que otras pueden tener un efecto mínimo o incluso negativo.
- Effect on operational performance (Impacto en el rendimiento operativo): Finalmente, esta columna evalúa cómo cada capacidad o proceso influye en el rendimiento operativo general de la organización. Algunas prácticas pueden resultar en una mejora menor o sustancial en el rendimiento operativo, mientras que otras pueden provocar una disminución sustancial, como se indica en el caso del desarrollo basado en tronco, que puede causar una disminución en el rendimiento operativo debido a la complejidad añadida.

En resumen según Dave Farley en la página 24 (DORA (DevOps Research and Assessment), 2023) nos indica lo siguiente:

Figura 9
Benefits of continuous delivery

Technical capabilities and processes	Effect on software delivery performance	Mediated through continuous delivery?*
AI	Minor decrease	No
Continuous integration	Minor increase	Yes, completely
Code review speed	Substantial increase	Yes, partially
Loosely coupled architecture	Minor increase	Yes, partially
Trunk-based development	Minor increase	Yes, completely

Nota: Adaptado de *Benefits of continuous delivery,* por (DORA (DevOps Research and Assessment), 2023)

El artículo destaca la esencia de la entrega continua (CD) en el desarrollo de software, resaltando la necesidad de mantener un software constantemente listo para su lanzamiento mediante un enfoque en la calidad. Se enfatiza la relación entre la calidad del software y la capacidad de proporcionar una retroalimentación rápida, indicando que métricas como la tasa de fallos en cambios y el tiempo de recuperación de despliegues fallidos son fundamentales para evaluar el rendimiento en la entrega de software. Además, se subraya el papel crucial de la CD como mediadora de muchas capacidades técnicas, facilitando una predicción más precisa del rendimiento en la entrega de software.

Asimismo, se destaca la importancia de la "releasabilidad" como un estándar esencial en el desarrollo de software, donde la CD emerge como un vehículo para garantizar cambios lanzables de forma rápida, segura y sostenible. Se argumenta que optimizar para mantener los cambios "releasables" constituye una optimización para alcanzar una definición contextual de calidad aceptable para el sistema. Sin embargo, el artículo también plantea interrogantes sobre el aparentemente bajo impacto de la integración continua y el desarrollo basado en tronco en el rendimiento de entrega de software, sugiriendo un interés en profundizar en este aspecto para comprender mejor su relación con la práctica de la entrega continua.

2.1.2. Análisis y discusión de resultados

Con base en los objetivos específicos establecidos, se realizó un estudio exhaustivo utilizando varios métodos de investigación, incluyendo el sistemático, histórico-lógico, análisis y síntesis, análisis documental y observación. Se investigaron las diferentes estrategias de automatización de pruebas empleadas en la Integración Continua para el desarrollo de aplicaciones web, así como las principales herramientas utilizadas en este contexto. Además, se valoró la eficacia y eficiencia de cada estrategia de automatización de pruebas en términos de detección de errores, cobertura de pruebas y tiempo de ejecución, recurriendo a documentación oficial y utilizando herramientas como los informes de DORA (DevOps Research and Assessment), análisis de documentos y listas de cotejo.

La investigación reveló una variedad de estrategias de automatización de pruebas utilizadas en la Integración Continua, mi persona se enfocó solo en dos de estas la automatización de pruebas Unitarias y la automatización de Pruebas de Integración. Se identificaron herramientas populares como JUnit, NUnit, PHPUnit, Selenium WebDriver y SoapUI, entre

otras, que son ampliamente utilizadas en el desarrollo de aplicaciones web para automatizar diferentes tipos de pruebas. Además, se encontró que la eficacia y eficiencia de cada estrategia de automatización de pruebas varían según diversos factores, como la complejidad del proyecto, el entorno tecnológico, los requisitos de calidad y los recursos disponibles.

El análisis de los resultados destacó la importancia de seleccionar la estrategia de automatización de pruebas adecuada según las necesidades y características específicas de cada proyecto. Se observó que las herramientas de pruebas unitarias como JUnit y NUnit son altamente efectivas para la detección temprana de errores y ofrecen una cobertura exhaustiva de código, mientras que herramientas de pruebas de interfaz de usuario como Selenium WebDriver y SoapUI son más adecuadas para validar la funcionalidad y la usabilidad de las aplicaciones web. Sin embargo, se identificaron desafíos en términos de tiempo de ejecución y mantenimiento de las pruebas, especialmente en proyectos de gran escala.

Una de las principales limitaciones encontradas durante la investigación fue la escasez de reportes o informes específicos sobre la automatización de pruebas en la Integración Continua. Aunque existen numerosos recursos y estudios sobre prácticas de DevOps, incluyendo la Integración Continua y la automatización de pruebas, encontrar información detallada y actualizada sobre estrategias específicas de automatización de pruebas dentro del contexto de la Integración Continua resultó ser un desafío.

Esta limitación dificultó la obtención de datos concretos y actualizados sobre las mejores prácticas, tendencias emergentes y casos de estudio relevantes en el campo de la automatización de pruebas en la Integración Continua. Como resultado, se requirió un enfoque más exhaustivo y multidisciplinario que incluyera la recopilación de información de diversas fuentes, como documentos técnicos, blogs especializados, foros de discusión y entrevistas con profesionales de la industria.

Aunque la información específica sobre nuestro tema de monografía no estaba directamente disponible a través de DORA, esta organización contribuyó de manera significativa al proporcionar estadísticas relevantes sobre la Integración Continua en el desarrollo de software. A través de sus investigaciones y análisis en el ámbito de DevOps, DORA ha ofrecido datos valiosos que contextualizan y respaldan nuestro estudio. Sus estadísticas sobre la frecuencia de implementación, la estabilidad de las versiones y otros

aspectos clave han proporcionado una base sólida para nuestro análisis y discusión sobre la Integración Continua y su impacto en el desarrollo de software.

En conclusión, el estudio proporcionó una visión completa de las estrategias de automatización de pruebas en la Integración Continua para el desarrollo de aplicaciones web, destacando la importancia de una evaluación cuidadosa de las herramientas y enfoques disponibles para lograr una implementación efectiva y eficiente de la automatización de pruebas en el ciclo de vida del desarrollo de software.

2.2. Conclusiones y Recomendaciones

2.2.1. Conclusiones

- La Integración Continua (CI) y la automatización de pruebas son prácticas fundamentales en el desarrollo de aplicaciones web, permitiendo una entrega más rápida y confiable del software al detectar errores de forma temprana y garantizar una alta calidad en el código.
- Existe una variedad de estrategias de automatización de pruebas disponibles, desde pruebas unitarias hasta pruebas de interfaz de usuario, cada una con sus propias ventajas y desafíos. La selección de la estrategia adecuada debe basarse en las necesidades específicas del proyecto y las características del equipo de desarrollo.
- Las métricas como la detección de errores, la cobertura de pruebas y el tiempo de ejecución son indicadores clave para evaluar la eficacia y eficiencia de las estrategias de automatización de pruebas en el marco de la Integración Continua. Sin embargo, es importante considerar también otros factores como la mantenibilidad de las pruebas y la facilidad de uso de las herramientas.
- La mejora continua es clave en el proceso de Integración Continua y la automatización de pruebas. Realizar evaluaciones periódicas de las estrategias y herramientas utilizadas, así como estar al tanto de las tendencias y mejores prácticas en el campo, permitirá optimizar continuamente el proceso de desarrollo de software.

2.2.2. Recomendaciones

- Priorizar la implementación de pruebas automatizadas en el marco de la Integración Continua como parte integral del proceso de desarrollo de software. Esto garantizará una detección temprana de errores y una mayor confiabilidad en las versiones del software producido.
- Mantenerse al día con las tendencias y las mejores prácticas en este campo garantizará la eficacia y eficiencia de las pruebas automatizadas.
- Recordar que en DevOps La comunicación efectiva y la colaboración entre los equipos son clave para el éxito en la implementación de estas prácticas.
- Una de las recomendaciones clave es realizar una investigación preliminar exhaustiva antes de elegir un tema para asegurarse de que exista la documentación necesaria que permita alcanzar los objetivos establecidos. Esto implica revisar la disponibilidad de información relevante, datos estadísticos, estudios de caso y documentos técnicos relacionados con el tema de interés. Al garantizar que haya una base sólida de documentación disponible, se aumenta la probabilidad de obtener respuestas claras y significativas a los objetivos planteados en la investigación. Esto también ayuda a evitar posibles obstáculos y limitaciones causados por la falta de información adecuada durante el proceso de investigación y análisis.

BIBLIOGRAFÍA

AppMaster. (24 de julio de 2023). *El papel de DevOps en el desarrollo web*. Obtenido de AppMaster: https://appmaster.io/es/blog/devops-en-el-desarrollo-web

Aranguren Velasquez, M. J., & Ruiz Pedraza, A. E. (2023). Incorporacion de la Metodologia DevOps en la construccion de Software de la Aplicacion Navega Seguro. *(Tesis de Licenciatura).* Universidad Catolica de Colombia, Bogota.

Autentia. (2023). *DEVOPS - GUÍA COMPLETA.* Valencia, España: Creative Commons Attribution ShareAlike 4.0 International (CC BY-SA 4.0).

Babbie, E. (2016). *Fundamentos de la Investigacion Social.* Orange, California: International Thomson Editores. S. A. de C. V.

Celi Parraga, R. J., Bone Andrade, M. F., & Mora, O. A. (2023). *Programacion Web Del Frontend al Backend.* Santo Domingo, Ecuador: Grupo AEA.

Cruz Gonzales, G. N., & Franco Calderon, J. A. (2023). Integracion y Despliegue Continuo con DEVOPS como cultura en empresas del sector TI Colombianas. *Rev. Ingeniería, Matemáticas y Ciencias de la Información*, 103-126.

DORA (DevOps Research and Assessment). (2023). *Accelerate State of DevOps Report 2023.* Obtenido de DORA: https://dora.dev/publications/

Estayno, M., Dapozo, G., Cuenca Pletch, L., & Greiner, C. (2009). Modelos y métricas para evaluar calidad de software. *Red de Universidades con Carreras en Informática (RedUNCI)*, 382-388.

FasterCapital. (10 de 03 de 2024). *Integracion continua automatizacion de versiones y pruebas*. Obtenido de FasterCapital: https://fastercapital.com/es/contenido/Integracion-continua--automatizacion-de-versiones-y-pruebas.html#Mejores-pr-cticas-para-pruebas-de-integraci-n-continua

Felipe Redondo, A. M., & Nuñez Cardenas, F. (2022). DevOps un vistazo rapido. *Ciencia Huasteca Boletín Científico de la Escuela Superior de Huejutla*, 6.

Fernandez, P. (5 de Junio de 2023). *Docker y microservicios*. Obtenido de Hiberus Blog: https://www.hiberus.com/crecemos-contigo/docker-y-microservicios/

Garcia Puebla, I. (28 de Diciembre de 2009). *soapUI: jugando con web services*. Obtenido de Adictos al trabajo, por Autentia: https://www.adictosaltrabajo.com/2009/12/28/introduccion-soap-ui/

Geek, R. (3 de Febrero de 2023). *Como realizar pruebas unitarias con PHPUnit*. Obtenido de RicardoGeek: https://ricardogeek.com/como-realizar-pruebas-unitarias-con-phpunit/

Giraldo Colorado, S. M., & Giraldo Plaza, J. E. (Julio-Diciembre de 2013). SISTEMA DE GENERACIÓN AUTOMÁTICA DE SCRIPTS DE EJECUCIÓN PARA PRUEBAS

UNITARIAS EN APLICACIONES WEB. *Revista Politécnica ISSN 1900-2351 Volumen 9 Año 9, Número 17*, 105-112.

Gonzalez Espinoza, S., & Bello Lara, R. (2016). Personalización del proceso de pruebas unitarias empleando la herramienta NUnit. *Serie Científica de la Universidad de las Ciencias Informáticas Vol. 9, No. 4*, 1-14.

Hall, T. (2024). *Prácticas recomendadas de DevOps*. Obtenido de ATLASSIAN: https://www.atlassian.com/es/devops/what-is-devops/devops-best-practices

Hiberus. (11 de 08 de 2023). *Las mejores herramientas para cada tipo de testing*. Obtenido de HIBERUS BLOG: https://www.hiberus.com/crecemos-contigo/las-mejores-herramientas-para-cada-tipo-de-testing/

Hristov, A. (2024). *ATLASSIAN*. Obtenido de Cómo las pruebas automatizadas hacen posible DevOps: https://www.atlassian.com/es/devops/devops-tools/test-automation#:~:text=La%20práctica%20de%20automatización%20de,y%20la%20experiencia%20del%20usuario.

Humble, J., & Farley, D. (2010). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation.* Reading, Massachusetts: Addison-Wesley Signature Series.

Humble, J., & Farley, d. (2011). *Continuous Delivery.* Massachusetts: Addison-Wesley.

Jakubiak, N. (6 de Diciembre de 2022). *Tutorial de JUnit: Configuración, escritura y ejecución de pruebas unitarias de Java*. Obtenido de PARASOFT: https://es.parasoft.com/blog/junit-tutorial-setting-up-writing-and-running-java-unit-tests/

Jet Brains. (2000-2024). *Pruebas automatizadas para CI/CD*. Obtenido de Jet Brains/Teamcity: https://www.jetbrains.com/es-es/teamcity/ci-cd-guide/automated-testing/

Lerna-Blasco, R. V., Murcia Andres, J. A., & Mifsud Talon, E. (2013). *Aplicaciones Web.* 28023 Aravaca (Madrid): McGraw-Hill/Interamericana de España, S.L.

Lujan Mora, S. (2017). *Programacion de aplicaciones Web: Historia, principios basicos y clientes web.* San Vicente, Alicante, España: Editorial Club Universitario.

openinnova. (21 de Octubre de 2022). *Diferencia entre Integracion Continua y Despliegue Continuo*. Obtenido de openinnova: https://www.openinnova.es/diferencia-entre-integracion-continua-y-despliegue-continuo/

Pachacuti Blanco, M. (2020). DevSecOps, Estado del Arte en el Contexto Boliviano. *Revista PGI. Investigación, Ciencia y Tecnología en Informática*, 72-75.

PARASOFT. (2024). *¿Qué son las pruebas automatizadas y por qué las necesita?* Obtenido de PARASOFT: https://es.parasoft.com/solutions/automated-testing/

PUPPET, CIRCLECI, SPLUNK. (2019). *2019 State of DevOps Report: Presented by Puppet, CircleCI, and Splunk.* Obtenido de Puppet + Circleci: https://www2.circleci.com/2019-state-of-devops-report.html

QAlified. (17 de 2 de 2011). *Las siete mejores prácticas de automatización de pruebas*. Obtenido de QAlified Building Quality: https://qalified.com/es/blog/mejores-practicas-en-testing-automatizado/

QAlified. (3 de Diciembre de 2022). *Introducción a Selenium Testing*. Obtenido de QAlified Building Quality: https://qalified.com/es/blog/introduccion-a-selenium-testing/

QAlified. (29 de 08 de 2023). *Pruebas de Integración: qué son, tipos y ejemplos*. Obtenido de QAlified Building Quality: https://qalified.com/es/blog/pruebas-de-integracion-que-son/

QAwerk. (10 de Octubre de 2023). *Métricas de pruebas de software más importantes*. Obtenido de QAwerk: https://qawerk.es/blog/metricas-de-pruebas-de-software/

Red Hat. (2022). *La integración y la distribución continuas (CI/CD)*. Obtenido de Red Hat: https://www.redhat.com/es/topics/devops/what-is-ci-cd#:~:text=La%20integraci%C3%B3n%20continua%20(CI)%20es,y%20distribuyen%20los%20cambios%20de

Rehkopf, M. (2024). *Pruebas de software automatizadas*. Obtenido de ATLASSIAN: https://www.atlassian.com/es/continuous-delivery/software-testing/automated-testing

Rengifo, J. P. (17 de 4 de 2023). *QUE ES DEVOPS, CUAL ES SU USO EN EL DESARROLLO DE SOFTWARE*. Obtenido de ITSOTWARE: https://itsoftware.com.co/content/que-es-devops-cual-es-su-uso-en-el-desarrollo-de-software/

Sharma, S., & Coyne, B. (2015). *DevOps para Dummnies.* Hoboken, NJ 07030-5774: John Wiley & Sons, Inc.

Tejada Yau, V. (16 de Abril de 2024). *Introducción el Framework de Pruebas Javascript Jasmine*. Obtenido de desarrolloweb.com: https://desarrolloweb.com/articulos/conociendo-jasmine.html

Valdez Valda, J. M. (29 de Marzo de 2022). *Testing de software: la importancia de automatizar los casos de prueba*. Obtenido de Encora: https://www.encora.com/es/blog/testing-de-software-la-importancia-de-automatizar-los-casos-de-prueba#:~:text=Entre%20los%20beneficios%20de%20la,funcionalidad%20rota%20en%20el%20build.